9-2015

inSense: A Variation and Fault Tolerant Architecture for Nanoscale Devices

John A. Frye

inSense: A Variation and Fault Tolerant Architecture for Nanoscale Devices

by

John A. Frye

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Engineering

Supervised by

Associate Professor Dr. Dhireesha Kudithipudi
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
September 2015

Approved by:

Dr. Dhireesha Kudithipudi, Associate Professor
Thesis Advisor, Department of Computer Engineering

Dr. Sonia Lopez Alarcon, Assistant Professor
Committee Member, Department of Computer Engineering

Dr. James Moon, Associate Professor
Committee Member, Department of Electrical Engineering

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:

inSense: A Variation and Fault Tolerant Architecture for Nanoscale Devices

I, John A. Frye, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

John A. Frye

Date

Dedication

This thesis is dedicated to my family, the greatest and most beautiful gift of all those I have to be thankful for. To name just a few, they include my parents, Carlota and John, my brother, Eric, my aunt, Martha, and my girlfriend, Megan.

Acknowledgments

Thanks to my committee members, Dr. Dhireesha Kudithipudi, Dr. James Moon, and Dr. Sonia Lopez Alarcon, for their expertise and support. Thanks also to Richard Tolleson and Dr. Roy Melton. Thanks to all the classmates and coworkers who have supported me in my efforts. Most of all, thanks to God, who provided me the wisdom and fortitude to complete this work.

Abstract

inSense: A Variation and Fault Tolerant Architecture for Nanoscale Devices

John A. Frye

Supervising Professor: Dr. Dhireesha Kudithipudi

Transistor technology scaling has been the driving force in improving the size, speed, and power consumption of digital systems. As devices approach atomic size, however, their reliability and performance are increasingly compromised due to reduced noise margins, difficulties in fabrication, and emergent nano-scale phenomena. Scaled CMOS devices, in particular, suffer from process variations such as random dopant fluctuation (RDF) and line edge roughness (LER), transistor degradation mechanisms such as negative-bias temperature instability (NBTI) and hot-carrier injection (HCI), and increased sensitivity to single event upsets (SEUs). Consequently, future devices may exhibit reduced performance, diminished lifetimes, and poor reliability.

This research proposes a variation and fault tolerant architecture, the inSense architecture, as a circuit-level solution to the problems induced by the aforementioned phenomena. The inSense architecture entails augmenting circuits with introspective and sensory capabilities which are able to dynamically detect and compensate for process variations, transistor degradation, and soft errors. This approach creates "smart" circuits able to function despite the use of unreliable devices and is applicable to current CMOS technology as well as next-generation devices using new materials and structures. Furthermore, this work presents an automated prototype implementation of the inSense architecture targeted to CMOS devices and is evaluated via implementation in ISCAS '85 benchmark circuits. The automated prototype implementation is functionally verified and characterized: it is found that error detection capability (with error windows from ≈30-400ps) can be added for less than 2% area overhead for circuits of non-trivial complexity. Single event transient (SET) detection capability (configurable with target set-points) is found to be functional, although it generally tracks the standard DMR implementation with respect to overheads.

Contents

List of Tables

List of Figures

Chapter 1

Motivation

Continued advances in transistor technology are accompanied by a plethora of issues and non-idealities with regard to device performance, reliability, and manufacturability. For digital designs, soft errors, process variations, and transistor degradation are critical issues which may require a change from traditional static, worst-case design paradigms. These problems, along with other future challenges, lend credence to a growing trend whereby systems must cope with non-uniform and unreliable devices in real-time rather than design time. This chapter explains the impetus for this research, while detailed consideration of the underlying physical phenomena and potential solutions is reserved for Chapter 2.

Overcoming wide-distribution process variations is a critical challenge for successful device scaling. CMOS process variations aggravate with scaling and must be addressed in order for scaling to remain beneficial [15]. Systematic variations can be accounted for to some degree through adjustments in the fabrication process and/or circuit design modifications. Random, intra-die process variations, however, cannot be accounted for using traditional static methods and must be considered from a worst-case paradigm [16, 17], potentially resulting in overly conservative and reduced operating frequencies. Post-silicon materials, which ITRS identifies as the next advancement over the next decade, will require means of reducing or dealing with increased ”variability of critical dimensions and statistical distributions” [18]. CNFETs are also prone to various types of device variation, including doping variation, CNT diameter variation, CNT density variation, and metallic

CNTs [19]. Graphene nanoribbon transistors (GNR) are expected to be even more susceptible to process variations than conventional MOSFETs [20].

Device reliability in advanced nodes may also be threatened due to a marked increase in the soft error rate (SER) contribution from logic [21, 22, 23], largely due to the decreasing efficacy of masking effects. Firstly, electrical and latching-window masking, which have traditionally prevented SETs in logic from affecting the system state, lose efficacy with scaling. Latching-window masking becomes less effective since SETs, which currently exhibit pulse widths between 200 ps to over 1 ns [22, 23, 24, 25], may be larger than the clock period of the system. Any such transient that arrives at the input of a flip-flop or latch is guaranteed to be captured. Some research, in fact, has suggested that pulse width increases with technology scaling [25]. Electrical masking also becomes less effective since fast logic gates do not attenuate the longer, high-amplitude transients characteristic of advanced devices; instead, they may propagate through a system indefinitely [22, 24].

Further aggravating the soft error rate is the increased probability of SET generation in combinatorial logic: supply voltage and node capacitance decrease with device scaling, and it follows that the charge threshold for SET generation decreases. This amount of charge is generally denoted Q_{Crit}, and previous research has found that the Q_{Crit} from 65 nm to 45 nm decreased by approximately 30% [26]. As a consequence, more common particle radiation with lower energy, such as neutrons resulting from cosmic-ray interactions, may cause transients of significance. More recent measurements in a 32nm process have not yet borne out extremely long pulse widths [27], but the measurements illustrate a trend of neutron-induced SETs becoming prevalent ($\approx$2.5x more frequent vs. alpha particle-induced SETs) in advanced technology nodes. Gill *et. al.* go on to say that additional decreases in Q_{Crit} or phenomena caused by different processes could "dramatically increase the significance of combinatorial logic SER in modern technologies".

Heavily modified processes [28] can serve to reduce the emission of alpha particles. The additional complexity and cost may prevent them from being widely available to all

design houses. In addition, while reducing the effective soft error rate due to alpha particles in logic, they do not address SETs caused by neutrons and protons. Finally, the number of points of failure in a system increases with device scaling due to increased design complexity and transistor density. Transistor degradation mechanisms, discussed in further detail later on, may further increase the susceptibility of a device to SET-generation [29].

Post-silicon materials and post-CMOS devices are also poised to suffer from single event effects (SEE). Group IV semiconductor materials have similar SEE sensitivities, and compound group III-V materials suffer from even greater sensitivity due to their increased internal gain [30]. Although PCM and MRAM-based devices are not very susceptible to particle interactions, required CMOS interface logic will exhibit the aforementioned vulnerability. Carbon nanotube FETs (CNFETs) are not expected to possess significant vulnerability to particle interactions, but this has not been exhaustively researched [30]. Furthermore, decreased reliability due to lower noise margins in concert with device crosstalk and increased susceptibility to environmental variations lends support to the necessity of dynamic error detection and/or correction in logic in the nanodevice era [20].

Scaled devices are also more susceptible to transistor degradation. The negative-bias temperature instability (NBTI) phenomenon is becoming an increasing concern in scaled devices [16]; power density increases with scaling, leading to high operating temperatures. In addition, [31] illustrates that the impact of NBTI increases as V_{DD} decreases. Although supply voltages decrease with scaling, they do not always decrease proportionally [3] in order to remain competitive in performance. This results in higher electrical fields in the device, which in turn increases the rate of degradation due to hot carriers.

The aforementioned issues are detrimental to the efficacy of future device scaling, and they all require an online, adaptive solution. These problems need to be solved for near-term and longer-term materials and devices. In fact, the 2013 edition of the International Technology Roadmap for Semiconductors report identifies a need for a ”shift to system level reliability perspective with unreliable devices” [32]. Consequently, these problems

demand an integrated, variation-tolerant architecture which accounts for variation sources in concert. An ideal architecture should be generic and applicable to any digital system, have minimal overhead, and should minimally impact the design of a system.

Chapter 2

Background

2.1 Device and Process Variations

Device and process variations describe deviations from planned device characteristics which are the natural result of imperfect fabrication technology and atomic-level differences [2]. From a digital designer's perspective, these variations fundamentally manifest as a variation in the threshold voltage of a given transistor and limit the maximum frequency at which each fabricated chip can operate. This results in a statistical distribution of operating frequencies around the target frequency. Figure 2.1 shows the random distribution of dopants within a 90nm MOSFET as well as the threshold voltage distribution for $\approx$ 3500 identically designed 90nm MOSFETs. A second example distribution is depicted in Figure 2.2; in this case, the frequency varies by 30% from the target, while the leakage current and - for a fixed supply voltage - power vary by more than 20 times. Process variations create a power/frequency/yield tradeoff; for example, yield can be increased by accounting for worst-case process variation in the operating frequency or voltage supply, leading to excessive limitation on performance and increased power consumption. In order to account for these distributions, designers must generally use some combination of worst-case design and frequency-binning. Variations are hierarchically classified as wafer-to-wafer variations, within-wafer (WIW) / inter-die variations, or within-die (WID) / intra-die variations. Typically, inter-die process variations directly affect the variation of the resultant frequency distribution [15, 16]. Intra-die process variations directly affect the

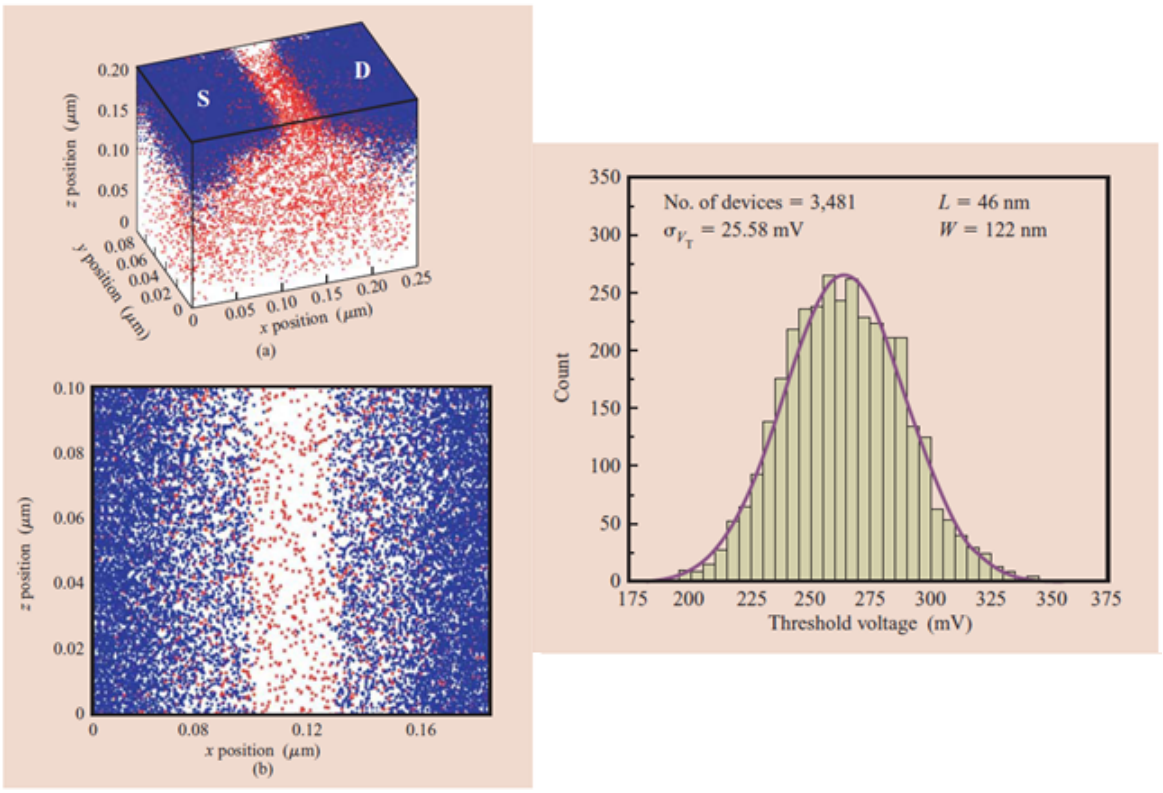


Figure 2.1: Figure of Randomly Placed Dopants and Threshold Voltage Distribution, 90nm [2]

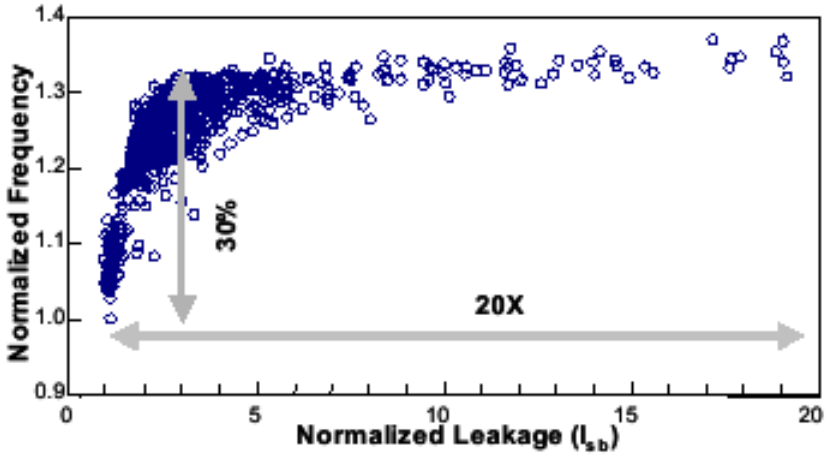


Figure 2.2: Power and Frequency Distribution due to Process Variations [3]

mean of the frequency distribution, and are thus of greater concern than inter-die variations.

In addition, process variations are classified as either systematic (*e.g.*, lens aberrations) or random (*e.g.*, random dopant fluctuations). Systematic variations can be mitigated (to some extent) with process adjustments, but random variations cannot be addressed without post-silicon tuning [17]. Post-silicon delay calibration can be accomplished through the application of *(a)* dynamic voltage and frequency scaling or *(b)* adaptive body-biasing.

a) Dynamic Voltage and Frequency Scaling (DVFS)

Dynamic Voltage and Frequency Scaling (DVFS), also referred to as Adaptive Supply Voltage (ASV), describes a technique wherein the circuit supply voltage and/or operating frequency are dynamically adjusted according to instantaneous power or timing demands. Classically, this allows for increased power efficiency by allowing the chip to run at a fraction of its high-performance conditions under reduced workloads. This scheme, however, can also help to compensate for both WIW and WID-type process variations. By raising the supply voltage to the system, timing constraints can be met in spite of variations by reducing the delay for critical blocks. Similarly, lowering the supply voltage to non-critical blocks serves to decrease dynamic power consumption. The frequency of the system is adjusted in tandem with the supply voltage in order to reap the benefits of reduced delay or to ensure timing requirements when power savings are desired.

b) Adaptive Body Biasing (ABB)

Adaptive Body Biasing (ABB) takes a complementary approach and uses the body effect to raise or lower a transistor's threshold voltage (V_T) in order to meet constraints. The main trade-off at play is that of increased drain current through devices, which decreases logic transition propagation delay while increasing static leakage current. Consequently,

ABB is partitioned into two operating modes: Forward Body Biasing (FBB) serves to increase the speed of transistors from their nominal value at the cost of increased power, while Reverse Body Biasing (RBB) increases delay and reduces power consumption. The effectiveness and overhead corresponding to each technique is determined by the granularity of application. Specifically, this refers to the number of supply/biasing blocks as well as the voltage range and resolution of each block. A system in which there is only one supply/biasing circuit with low voltage resolution would have little overhead. However, it would have limited efficacy since the entire circuit would be adjusted with respect to the worst process variation effect. Such a system would be able to account for some level of inter-die process variation but would not be able to accommodate intra-die process variations. By adding a unique supply/bias generator for each major circuit block in a system, a reasonable balance between effectiveness and overhead can be achieved. This configuration was successfully implemented in [4] and was shown to be quite effective. Figure 2.3 illustrates that the variation after application of the intra-die ABB technique is a third of that of the inter-die technique, and the power consumption and operating frequencies are more optimal.

The research in [33] illustrates that the difference between DVFS and ABB in their ability to compensate for process variations is negligible; however, the authors make the point that DVFS costs less than ABB. This is due to the fact that ABB requires an additional distribution network for the body-biasing voltages, while DVFS leverages the existing power-supply network. However, [4] and [34] show that the total area overhead associated with the bias generators and distribution network is only approximately 2-4%. Aside from the speed-up of devices which violate the delay requirement, ABB and DVFS can be used to obtain power savings from devices exceeding the delay requirement. With ABB, devices exceeding the frequency target can be reverse-biased in order to reduce static power dissipation. DVFS can be used to lower the supply voltage for devices exceeding the frequency requirement, which serves to reduce dynamic power consumption.

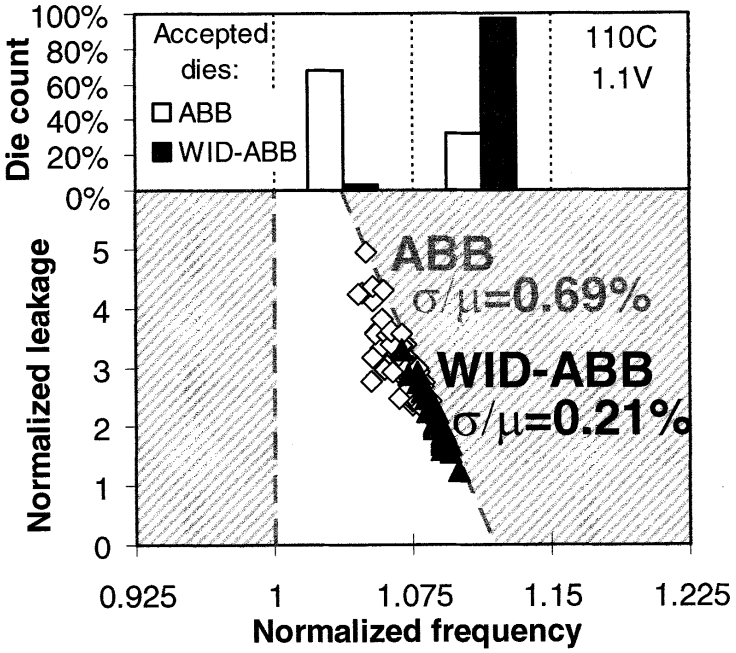


Figure 2.3: Die-to-Die and Within-Die ABB Comparison [4]

Consequently, ABB is more attractive for scaled devices due to the fact that static power dissipation is becoming the primary power concern. However, the authors in [34] show that DVFS and ABB can be combined in order to achieve both static and dynamic power savings.

2.2 Soft Errors

A soft error describes a corrupt calculation or signal resulting from some type of transitory effect - *i.e.*, an error caused by a fault other than a permanent hardware defect. Depending on the sequence of operations, the erroneous datum may or may not result in the corruption of the system's state. While soft errors can have many causes including random noise or signal integrity issues, they primarily manifest through electrical singularities introduced by incident particle radiation. Energetic atomic and subatomic particles (*e.g.*, heavy ions, protons, and neutrons) cause varying effects in silicon, and the radiation environment

determines which effects are preeminent. For example, in low earth orbit (LEO) environments, trapped protons are of principal concern, but heavy ions from cosmic rays cause the most problems in geo-synchronous orbit[35]. For terrestrial applications, alpha particles from impurities in packaging and secondary particles, mostly high-energy neutrons, generated by the interaction of cosmic rays with the earth's atmosphere dominate[6].

Particle radiation effects can be segregated into two groups for analysis - slow, lifetime degradation effects (*e.g.*, ionizing and displacement damage), and effects caused by a discrete particle interaction, typically referred to as a single-event effect (SEE). The former is generally only of interest in high radiation environments, while the latter can cause significant problems in both terrestrial and space applications. SEE phenomena can be further sub-divided into permanent, device-damaging effects, such as single-event latchup, single-event burnout, or single-event gate rupture, and effects which do not damage devices but may temporarily interfere with error-free circuit operation.

When a particle such as a heavy ion passes through the drain-depletion region of a FET, it transfers energy to the ions in the region in correlation with the stopping power action of the material. This energy transfer is characterized as the linear energy transfer (LET), and the amount of transferred energy depends on the particle, its angle of incidence, and the semiconductor material properties. The transferred energy promotes carriers in the depletion region from the valence band into the conductive band, leaving a collection of electron-hole pairs in the radiation trail as is depicted in Figure 2.4. The drain depletion-region is the most sensitive, as it may have a very large electric field due to reverse bias. This electric field sweeps away the freed carriers, resulting in a current. A sufficiently large current generates a transient (erroneous) signal, called a single-event-transient (SET). Figure 2.5 contains a depiction of the resultant transient, illustrating both the initial high-speed drift collection of the generated electron pairs and the slower tail created from the diffusion portion. More complex nuclear and spallation interactions (both elastic and inelastic) with protons and neutrons can also occur [6].

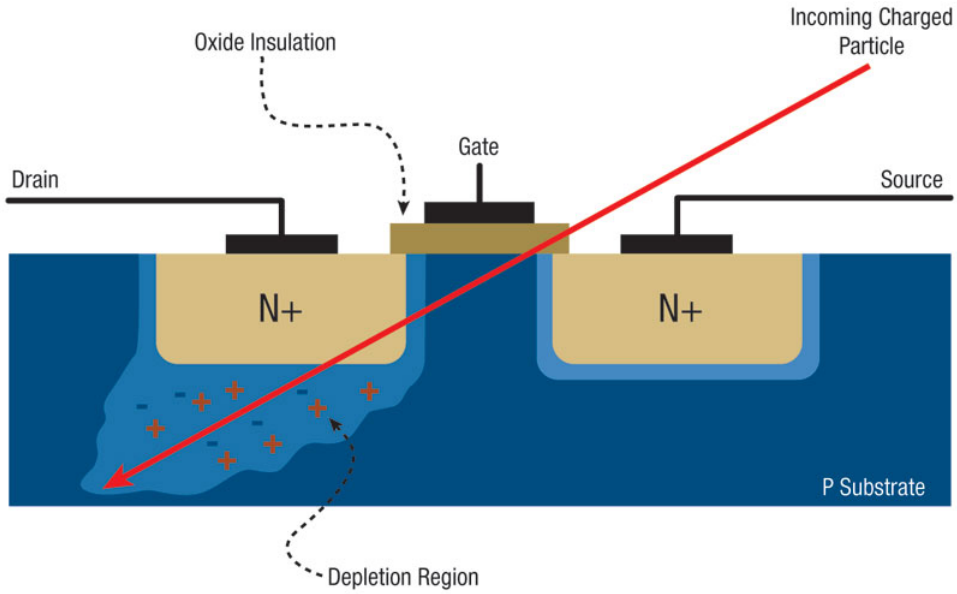


Figure 2.4: Energetic Particle Interaction with Device [5]

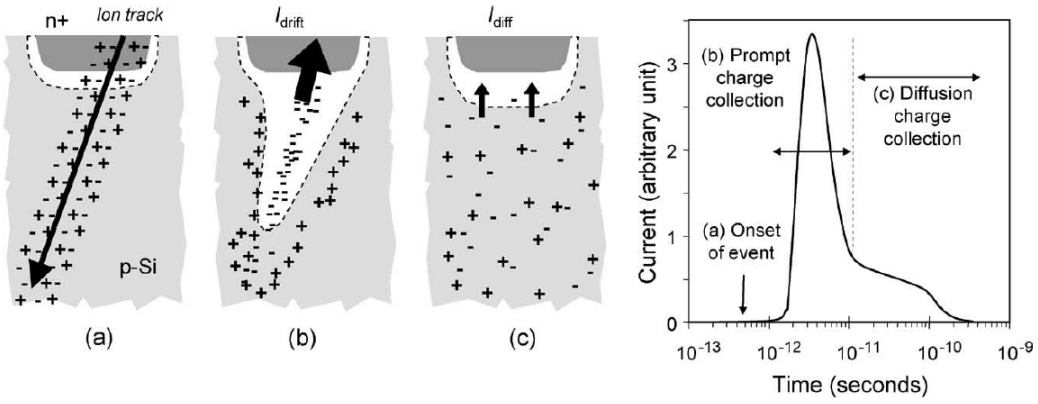


Figure 2.5: SET Current Pulse Characteristics from Ion Interaction [6]

Traditionally, the predominant soft error mechanism has been direct interaction of an energetic particle with a memory element such as a flip-flop, latch, or RAM cell[21]. If such an interaction causes enough charge (Q_{Crit}) to accumulate in a sensitive memory element node, a bit-flip can occur due to the positive feedback design of a memory element. The industry jargon for this type of soft error is referred to as a single-event upset (SEU). Error-correction codes (ECC), which are cost-effective in the context of regular, dense structures such as memory, have been and continue to be used for protection against bit-flips. Radiation-hardened designs [36, 37, 38] for memory elements, including latches and flip-flops in logic, can be used to reduce SEU susceptibility. Alternatively, SEUs in logic and memory can potentially be reduced through the use of radiation-hardened processes [28] at substantial cost.

A second SEU mechanism entails the interaction of an energetic particle with a node in combinatorial logic. An SET generated in a logic gate node can cause an SEU if it is latched, potentially corrupting the state of the system. Generally, SEUs caused by SETs in logic have occurred far less frequently than SEUs in memory. This is due to the fact that transient signals can be electrically, logically, or latching-window masked as they propagate through a circuit. Electrical masking describes the attenuation of transients due to the electrical characteristics of logic gates, while logical masking occurs when a transient is masked by another input to a logic gate. Latching-window masking occurs when a transient propagates to a latch during its opaque-phase. Since logic is not uniform like memory, ECC is not an efficient technique for mitigating SETs. However, there are a number of mitigation techniques which utilize either *a) temporal redundancy* or *b) spatial redundancy* to detect or mask SET errors [39, 7, 40, 41, 42, 8, 43].

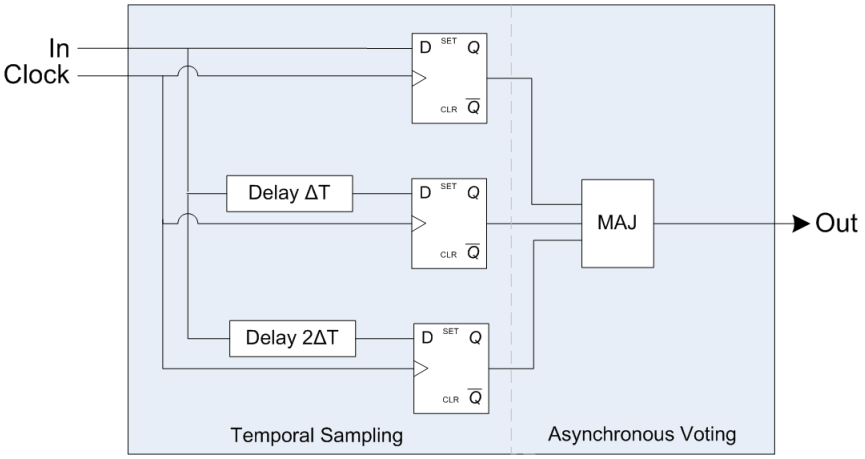


Figure 2.6: Temporal Sampling Technique [7]

a) Temporal Redundancy

One of the most popular and effective temporal redundancy-based techniques is temporal sampling [7], in which a latching element samples a signal multiple times in order to detect or mask an SET. The temporal sampling technique is illustrated in Figure 2.6. In order to avoid latching an incorrect value, the delay between each sample must be greater than the pulse width of the transient; hypothetically, this could pose a large frequency constraint on the system in cases where transients exhibit long pulse widths.

Another time-based technique involves error detection through redundant computations. A particularly efficient version of this involves executing two or more copies of each thread in order to detect and correct errors; this is often called redundant multithreading (RMT) and takes advantage of the thread-level parallelism (TLP) inherent in simultaneous multithreading (SMT). In this configuration, information can be shared between threads in order to speed up computation. The performance overhead with this technique is relatively low at approximately 10-30% depending on the specific implementation [40, 41, 42]. From a design paradigm, however, RMT is not ideal in that it requires hardware support and thus specifically impacts the design of a system.

b) Spatial Redundancy

The most fundamental spatial redundancy technique is modular redundancy, in which the hardware is replicated in order to redundantly execute instructions in parallel. This is usually implemented in the form of dual modular redundancy (DMR), in which two modules serve to detect errors, or triple modular redundancy (TMR), in which three modules mask errors through the application of majority voting. In their simplest form, these techniques are not a viable option as they have respective area overheads of 100% and 200%. However, a targeted application in which only the nodes most likely to generate a transient are replicated could be cost-effective.

One study [8] proposes characterizing the susceptibility of a node to creating an SET by its Q_{Crit}, the minimum charge an energetic particle must transmit to a node in order to generate a transient pulse. This definition implies that a node with a higher Q_{Crit} is less likely to generate a transient pulse than one with a lower Q_{Crit}. In the study, several circuits were analyzed according to this metric. It was found, as shown in Figure 2.7, that the susceptibility of circuit nodes to SET generation can vary by over an order of magnitude.

Consequently, only a subset of circuit nodes needs to be replicated in order to provide effective SET detection capabilities. This scheme is illustrated in Figure 2.8. In accordance with this finding, a greedy, area-based duplication algorithm is presented in [8]. This technique allows for variable levels of SET detection coverage depending on how much area overhead can be afforded; in the circuits studied, approximately 50-70% coverage can be obtained for 20% area overhead, while approximately 85-95% coverage can be obtained for approximately 50% area overhead. A disadvantage to partial duplication is that it is a detect and recovery scheme, and as such incurs additional area, performance, and power overhead due to the need for recovery logic. An alternative is to use partial triplication with majority voting, as is described in [43]. This solution masks errors as they occur and is completely transparent from a design point of view. However, the area overhead is much

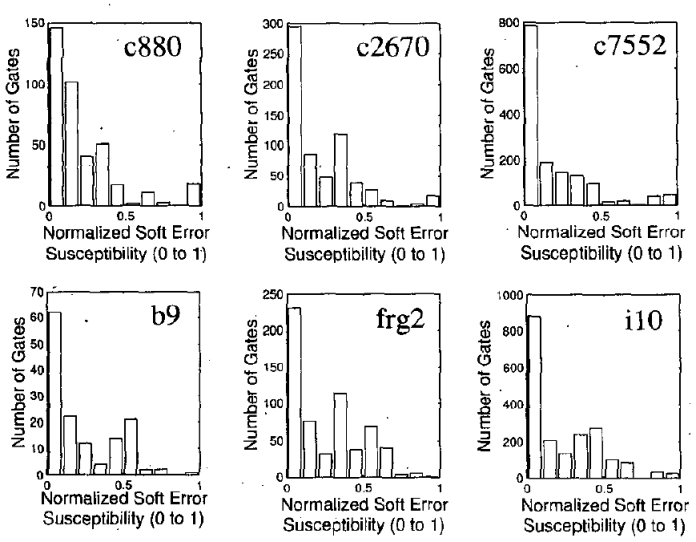


Figure 2.7: Soft Error Susceptibilty of Different Circuits [8]

greater and is typically over 100%.

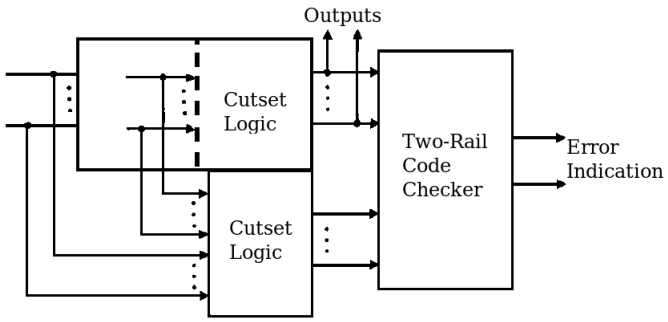


Figure 2.8: Partial Duplication Technique [8]

2.3 Transistor Degradation Mechanisms

Another limiting factor to device scaling is decreasing lifetime due to device degradation. These degradation mechanisms include electromigration, hot carrier effects, negative bias temperature instability (NBTI), and time-dependent dielectric breakdown (TDDB). Some of these degradation mechanisms cause a slow wearout of the device (*e.g.*, NBTI, hot carrier effects) which decreases lifetime when defined by a threshold voltage target; others, including electromigration and TDDB, cause hardware failure.

The severity of these mechanisms are temperature-dependent, and some, like hot carrier effects, are exacerbated under high electric fields. Figure 2.9 depicts the hot carrier injection degradation mechanism, whereby high electric fields in the channel result in impact ionization, which in turn causes carriers to disrupt or get trapped into the device oxide. The result of this is a threshold voltage increase. These can easily occur in scaled devices due to thinner gate oxides, increasing power densities, and the non-ideal scaling of supply voltages. Wearout mechanisms which affect device performance in terms of threshold voltage can be viewed as another element of variability to be compensated. In this case, ABB or DVS can be used for the purpose of extending device lifetime.

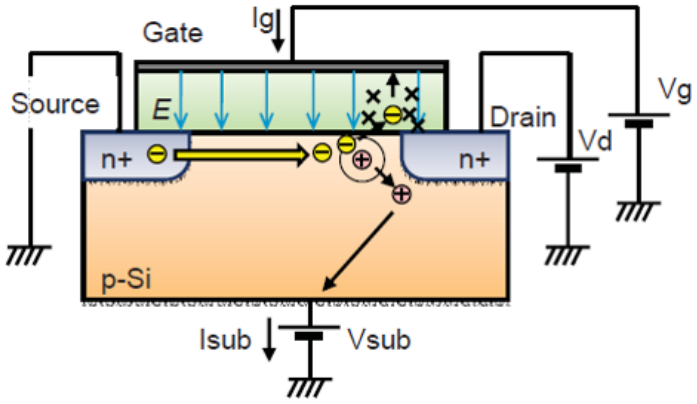


Figure 2.9: Depiction of Hot Carrier Injection Degradation [9]

2.4 Summary

Advanced technology nodes are subject to a plethora of issues challenging device performance, power consumption, yield, reliability and lifetime. Random, within-die process variations make the worst-case design paradigm untenable when pushing the performance envelope in advanced nodes, and device sensitivity to SEUs as a function of SETs in combinational logic looks to become a pertinent issue. Transistor degradation mechanisms also threaten to significantly reduce device lifetime.

Chapter 3

Related Work

There are existing reliable and variation-tolerant architectures targeted for scaled devices, including Razor II [11], Bulletproof [12], StageNet [13], and NanoBox [44]. These architectures have all focused on soft errors and/or silicon defects, and some even provide benefits in terms of performance and power; however, none have addressed the sources of variation in concert.

This research was inspired by the elastIC concept [10], which is an architecture designed to mitigate many of the problems associated with device scaling. In essence, elastIC embodies an adaptive architecture rather than a standard (static) architecture which is limited by worst-case design philosophy. The envisioned architecture entails a large-scale multi-core system of processing elements (PEs) with a central diagnostic and processing unit (DAP). A PE may consist of a simple processor, a BIST unit, and one or more sensors. Periodically, the DAP takes PEs offline for a health assessment through the use of automatically generated test patterns, and PEs are tuned and healed as necessary. The natural redundancy of a multicore system is exploited in the case of a hardware failure. The DAP is also capable of adjusting the power/performance trade-off of PEs on the fly dependent on instantaneous system need. Refer to Figure 3.1 for a depiction of elastIC. elastIC is a compelling concept, but it remains an architecture philosophy or blueprint; it is a valid concern whether or not such an architecture is achievable with low-enough overhead to justify its use.

The Razor II architecture utilizes a novel flip-flop with error detection capabilities [11]

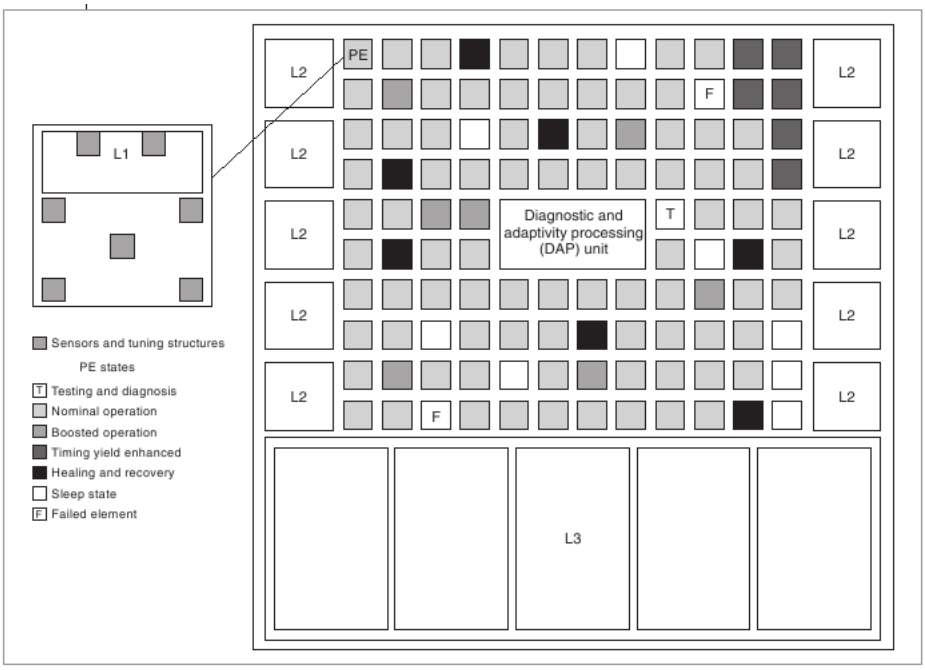


Figure 3.1: elastIC Adaptive Architecture [10]

in order to detect timing errors and SETs. Razor II designates a portion of the flip-flop's latching window as an error-detection window; any transitions during this period are assumed to be a fault. The Razor II schematic and timing diagram are shown in Figure 3.2. The error-detection provided by the Razor II flip-flop provides two benefits. The main benefit is the ability to detect timing errors; this allows for power savings, since a system with this capability can be run at a better-than-worst-case supply voltage. Secondly, transients or glitches which occur during the error-detection window can be detected by the Razor II flip-flop; this, in effect, is a more compact form of the temporal sampling technique. The design is effective in detecting SETs with small pulse widths, since only a small time-window is necessary in this case. Transients with large pulse-widths, however, will require large error detection windows; while the authors claim that a transient occurring during the opaque phase of the latching is benign, long pulses could easily begin during

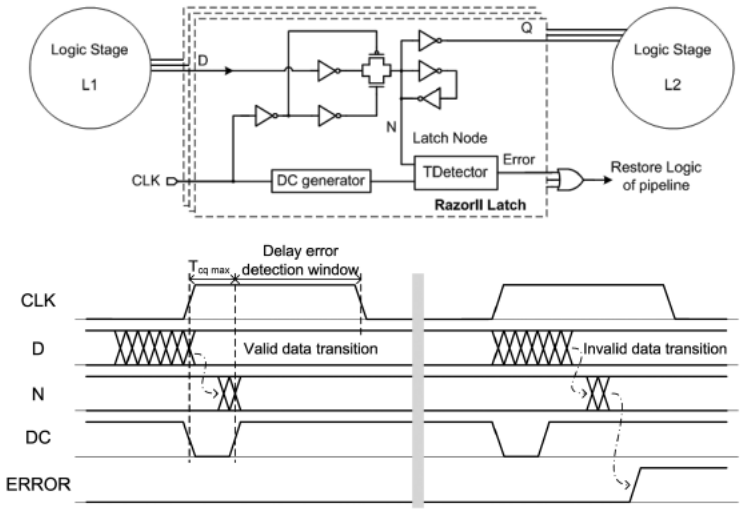


Figure 3.2: Razor II Architecture and Timing Diagram [11]

an opaque phase and last for several cycles, causing undetected errors. In the case of transients with pulse widths from 600 ps to over 1 ns, which are not uncommon in the 90 nm technology node, the corresponding error-detection windows could potentially impose a large frequency constraint on the system. As a result, the Razor II architecture does not efficiently provide protection against the large and steadily growing transient pulses that will affect scaled devices.

Another resilient architecture is the Bulletproof architecture, proposed in [12] and illustrated in Figure 3.3. Bulletproof utilizes small, specialized BIST checkers to validate the operation of each pipeline stage in a microprocessor. Validation occurs during pipeline stalls or idle time. If all hardware passes verification, the system continues to function normally; if a test fails, the hardware is assumed to be faulty. The offending hardware is disabled, and the system is rolled back to a previous checkpoint state. There is also a method proposed for mitigating soft errors; the proposed technique is based on temporal sampling, and as such, does not provide protection against the long transient pulses that scaled devices will be prone to. The Bulletproof architecture provides a novel, low-overhead design for detecting hardware faults and silicon defects, but does not account for

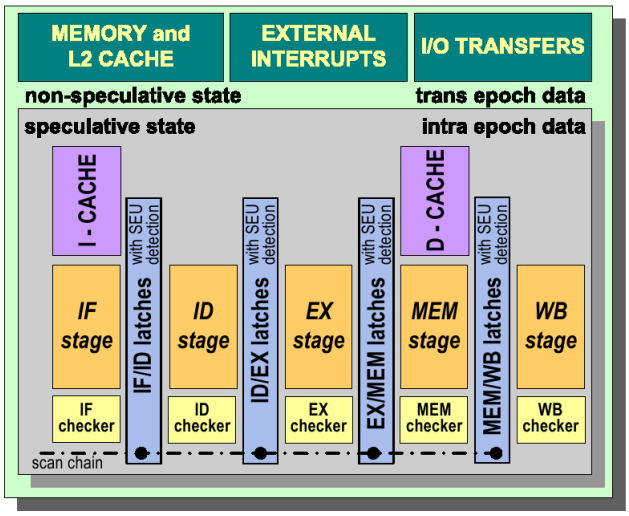


Figure 3.3: Bulletproof Architecture [12]

device variability.

The StageNet chip multiprocessor (CMP) architecture [13] is essentially a finer-grained implementation of a CMP than traditional designs; rather than have multiple independent cores on a processor, the StageNet architecture calls for a network of loose pipeline stages, connected by a configurable crossbar between each set of stages. In a normal CMP architecture, the method of tolerating a hardware fault would be to disable the entire parent core; the StageNet architecture, however, is far more efficient and would need to disable only the faulty pipeline stage. This is illustrated in Figure 3.4. The StageNet allows for a considerable boost in performance, by as much as 40-50% [13]. While the StageNet architecture is clearly robust against hardware faults, it does not solve the problems posed by soft errors, process variations, transistor degradation, or environmental degradations.

The architecture proposed in the NanoBox project [44] involves making each element in a circuit self-correcting. The implementation of this idea involves utilizing LUTs for all logic functions and using error-correcting-codes in order to automatically detect and correct errors. While this approach does allow for dynamic error correction, the area overhead

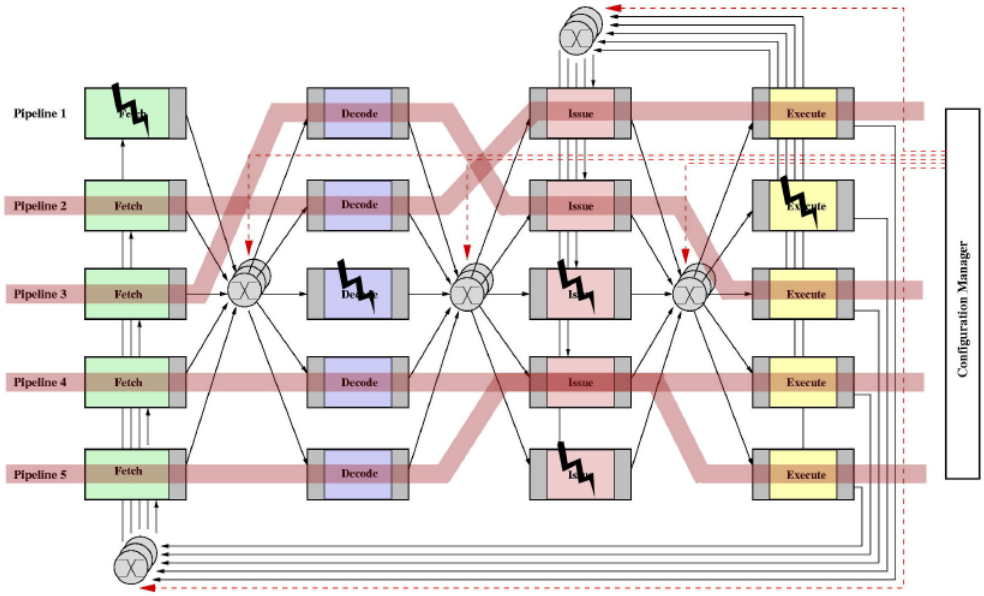


Figure 3.4: StageNet CMP Architecture [13]

is staggering at approximately 900%. This proposal may be feasible for future technologies which offer greater density but is certainly not justified for use with CMOS.

In summary, the need to define a dynamic, variation-tolerant architecture has been recognized by many, and the task remains to define an architecture that both comprehensively addresses the issues posed by non-ideal devices and is practical to implement from the perspective of circuit overhead and design effort.

Chapter 4

Proposed inSense Architecture

The inSense Architecture aims to address the shortcomings of unreliable circuit elements through the use of introspective sensing. By employing online delay-tuning and error detection, a circuit can become a "smart" circuit and dynamically correct for deviations in operation. This directly addresses intra-chip, wide-distribution process variations which threaten to limit operating frequency, soft errors, and transistor degradation. Conveniently, these schemes may also help to alleviate signal integrity issues caused by power-supply glitches and noise. Figure 4.1 depicts an example application of the inSense architecture in a simple RISC microprocessor. Like an animal's exoskeleton, the inSense architecture

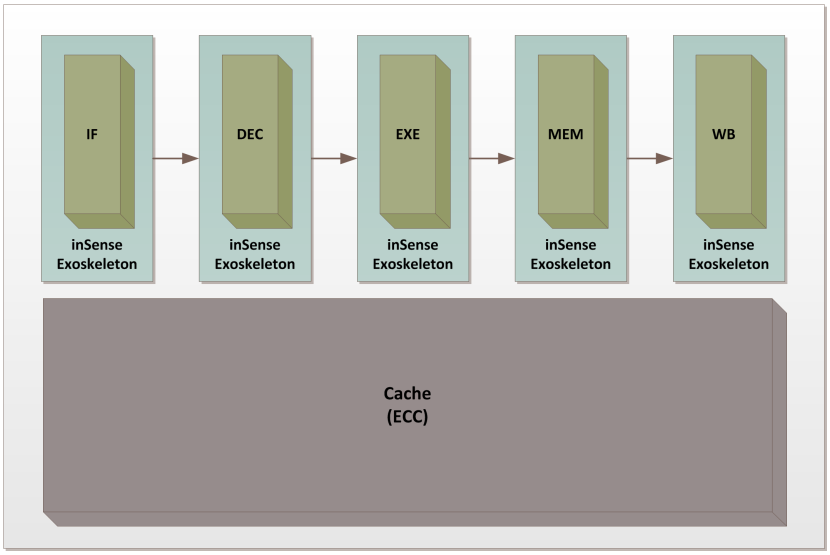


Figure 4.1: Microprocessor Augmented with inSense Architecture

entails the addition of protective and sensory capabilities to a host system. Consequently, the fundamental component of the architecture is termed the “inSense Exoskeleton”. Each inSense Exoskeleton comprises three main elements, as shown in Figure 4.2: i) the delay monitor unit (DMU), ii) the variation control unit (VCU), and iii) the Soft-Error Detection Unit (SDU). The DMU is able to detect the effect of process variations, transistor degradations, and environmental variations; these are detected indirectly through the measurement of circuit propagation delay. The DMU, in turn, signals the VCU to tune the circuitry upon detection of excessive delay. The SDU serves to detect and/or mask SETs in logic and may also help to mitigate other glitches and soft errors.

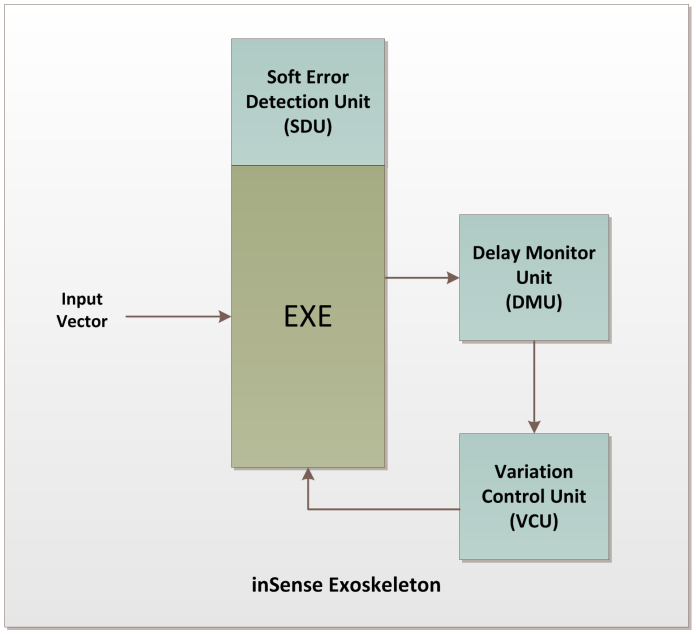


Figure 4.2: inSense Exoskeleton

As one might imagine, there is an efficacy/overhead trade-off depending upon the granularity with which the exoskeleton is applied to the system. Applying the exoskeleton once at the top level to cover the entire design would result in the least overhead, but the worst-case intra-die process variations would dominate the chip. In contrast, applying an exoskeleton to each logic gate would allow for very fine-grained control for the VCU but

would be intractable. A sensible solution is one in which an inSense exoskeleton is applied to each of the major circuit blocks in a system, striking a good balance between overhead and delay compensation granularity.

An objective of this research is to propose an architecture that can be automated in order to avoid additional complexity in the design process. Figure 4.3 illustrates the prototype inSense implementation flow. After the target design is synthesized, the resulting netlist and reports are fed as inputs into the DMU insertion process. The DMU-augmented netlist then undergoes the SDU insertion process, pursuant to the desired area overhead constraint. The design is then ready for implementation in accordance with the standard EDA flow. The DMU, SDU, and associated implementation flows are described in detail in the following sections.

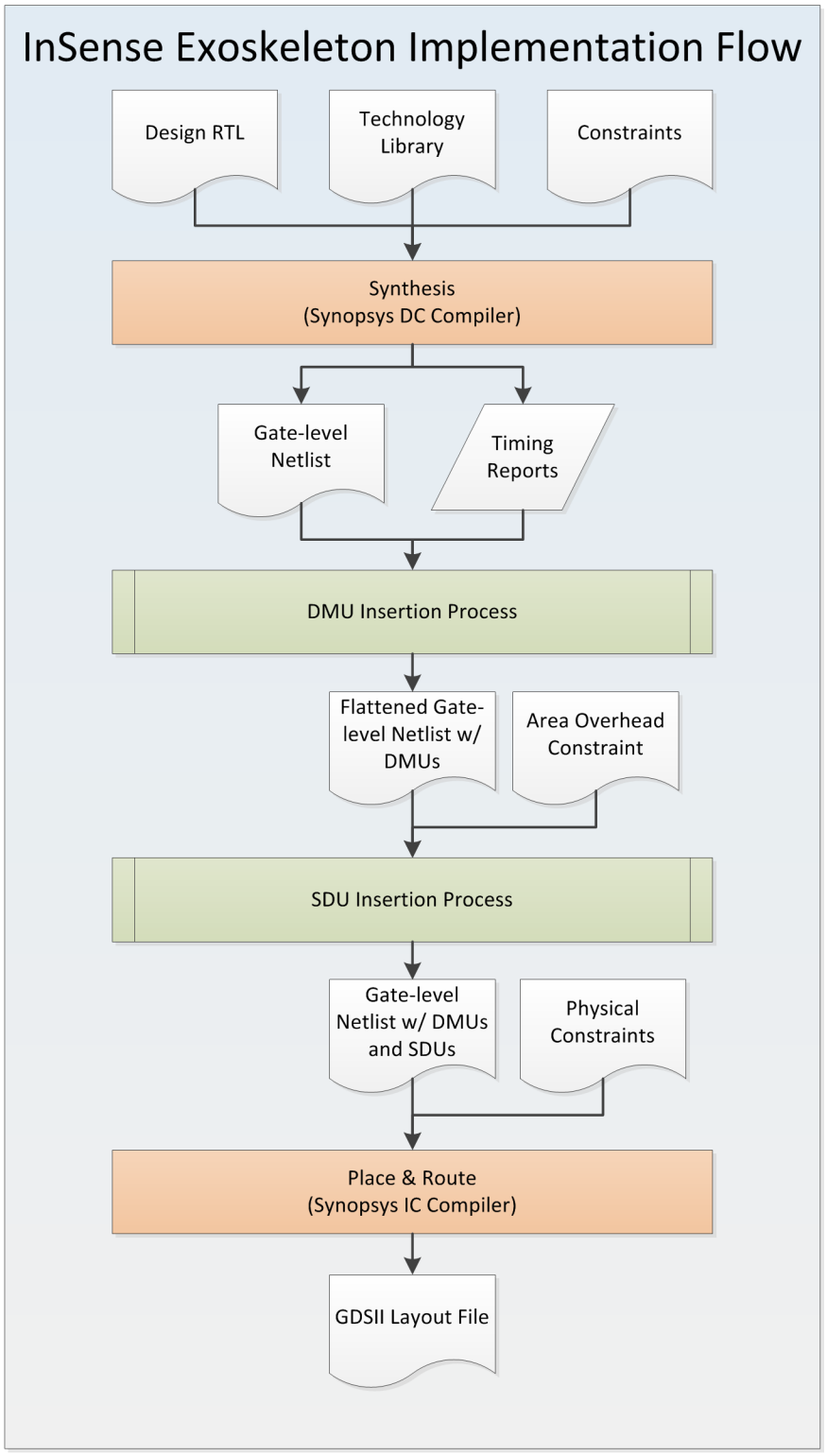


Figure 4.3: inSense Implementation Flow: RTL to Layout

4.1 Delay Monitor Unit

The delay monitoring unit is conceptually similar to the models proposed in [34] and [45] but differs in that it will measure the delay of the actual logic. This allows for transistor degradation, temperature variation, and supply voltage variation to be accounted for in addition to process variation. It also negates the need for replicas of the critical path. Idyllic delay-measurement characteristics include:

1. Large delay detection window
2. Small delay detection granularity
3. Circuit timing transparency - *i.e.*, little to no effect on timing closure of measured circuit
4. Data path metastability prevention, *i.e.*, having the ability to detect delays before they violate timing on the datapath
5. Quantitative delay measurement
6. Minimal circuitry and impact on the area footprint of the design

An ideal delay monitor would be capable of detecting a span of circuit delays while incurring minimal timing, area, and power overhead. Additionally, an ideal DMU would avoid any chance of metastability on the data path; circuitry designed to tolerate faults should not itself contribute problems. The availability of quantitative delay information could allow for more sophisticated status reporting and variation compensation. Different DMU designs, detailed in the following sections, trade off between these characteristics.

4.1.1 BIST-Based Design

The first delay monitor candidate, the BIST-based DMU, consists of a test-vector generator, a verification unit, and a high-speed counter. A simplified diagram of this implementation is depicted in Figure 4.4. As in a traditional BIST circuit, the test input is fed to the unit under test, and a known good output is compared against the circuit output. In this case,

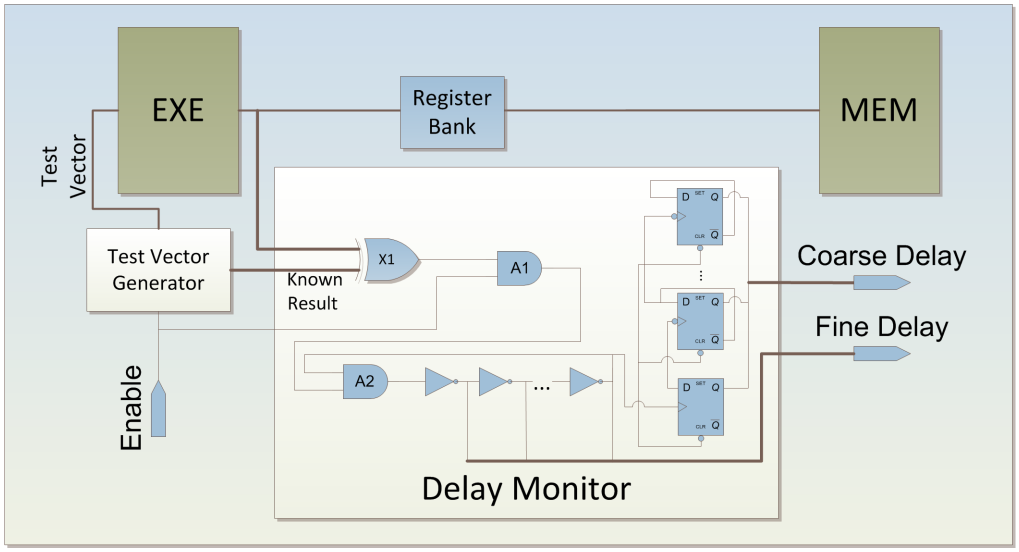


Figure 4.4: BIST-Based Delay Monitor

the high-speed counter is used to quantify the propagation delay between the test-vector input and the correct circuit output. The counter implementation includes both a ring oscillator (RO), which can be tapped to act as a fine-grained counter, and a binary counter (clocked by the ring oscillator) to act as a coarse-grained counter. This approach provides the benefit of a quantified delay measurement, which can be used by the variation control unit for immediate, proportional compensation. The delay measurement does not interfere with normal operation of the circuit, and it is also capable of measuring multi-clock cycle delays.

The circuit works as follows: first, the enable signal is asserted by control logic, which must clear the flip-flops and sensitize the feedback path of the ring oscillator through gates A1 and A2. With the feedback path enabled, the ring-oscillator can oscillate and clock the flip-flops every time the inverter chain is traversed. This time keeping continues until the logic circuit output matches the known output corresponding to the test vector, causing a '0' on the output of gate X1 and removing the feedback path from the RO.

Design and use of the BIST-based monitor can be difficult as the capacitive loads of

the fine-delay tap and the sequential element for the coarse-delay counter significantly increase the propagation delay of the ring oscillator, reducing the resolution of the monitor; sub-clock cycle delay measurement may be difficult or impossible depending on the amount of pipelining in the system. Careful design may be able to help alleviate some of these issues; for example, the resolution of the counter could be improved by using DDR flip-flops. Another issue complicating the design is the specific mechanism for capturing the state of the ring oscillator taps at the moment the delay measurement is disabled. If transmission gates were to be included between each of the inverters in the RO, the output of each inverter could be isolated at the end of measurement, and either physical or parasitic capacitances could store the ring oscillator state for sampling by a register. Finally, the requirement for a test vector generator for each logic block as well as an output comparator could result in high area overhead. This overhead can be minimized by hand-selecting test vectors which pass through the critical paths in each logic block. Alternatively, the circuitry overhead could be reduced through re-use of existing BIST circuitry required by a given circuit design.

4.1.2 Data-Delay Design

A second design, illustrated in Figure 4.5, is based on the temporal sampling concept, which has previously been used for detecting both timing errors and SETs. A delay chain is employed in order to detect delays on input signals. The secondary sequential element, connected to the delay line, serves as a delay warning mechanism by recording erroneous values before the primary element data path is affected. Comparison of the two stored values via the XOR gate indicates the presence of excessive delay to the system. Although only a simple binary delay measurement is provided, the design is simple to implement and conceptualize. This design could potentially result in metastability on the DMU error output, although this is not an issue of great concern; at worst, this could cause a borderline delay condition to fail to flag an error, and registering the output serves to synchronize

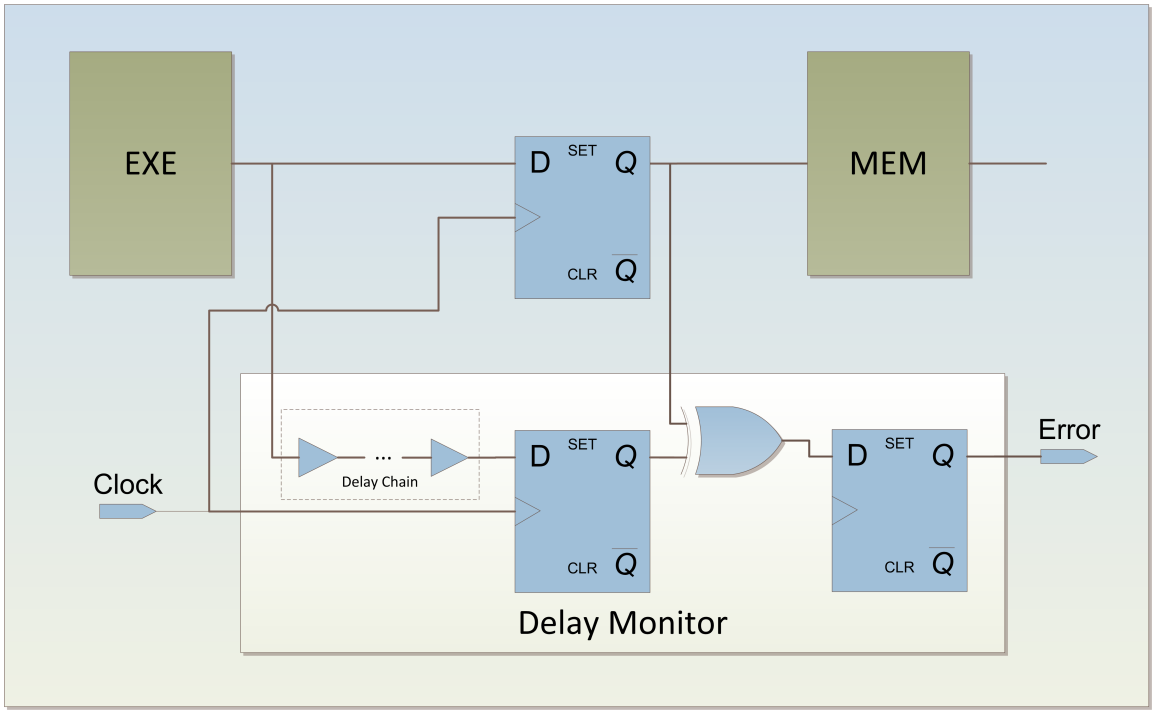


Figure 4.5: Data Delay-Based Delay Monitor

it and mitigate this possibility. The reliable delay detection window is expressed via the following equation:

$$slack_{d2} + hold_{d2} < T_{rel\_window} < slack_{d1} - setup_{d1} \tag{4.1}$$

In the above equation, the symbol *d2* represents the sequential element present in the delay monitor, while the symbol *d1* represents the flip-flop present in the data path. If the delay falls on the short side of this window, the reliability of the detection mechanism may be compromised, as the setup/hold time requirements of the secondary sequential element might be violated; while this behavior is non-ideal, it is likely acceptable for the reasons mentioned previously. If, however, the delay falls on the high side of the window, the primary sequential element could become metastable, compromising the integrity of the data path. From (4.1), it is clear that the difference between the main data path slack and the delay line slack should be maximized in order to optimize the error detection window. In other words, the length of the delay chain is proportional to the desired delay detection resolution. A delay-chain with delay equal to the slack time on the data path will allow the delay monitor to detect delays just greater than the hold time requirement of the register; shorter delay chains will reduce the overhead associated with the delay-monitor, but smaller delays may go undetected. In addition, this type of delay monitor may incur a timing penalty if the desired error window is larger than the available slack for a given data path.

4.1.3 Clock-Delay Design

Figure 4.6 exhibits an alternative configuration in which the sampling delay is applied to the secondary flip-flop's clock input. This scheme eliminates any circuit timing impact as it allows for delay detection for time *T* after the clock edge, where *T* is the sampling delay. One convenience of this approach, as outlined in [12], is that an additional error detection window of half a clock period can be achieved by using an inverted clock signal as the

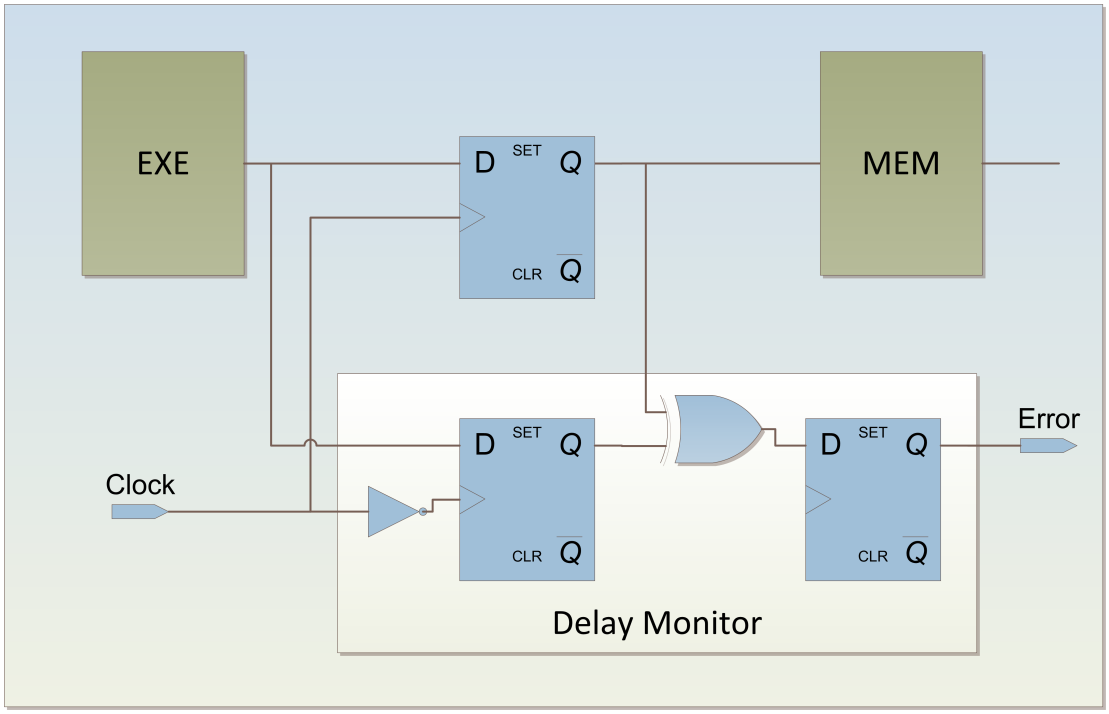


Figure 4.6: Clock Delay-Based Delay Monitor

sampling delay, lowering the circuitry overhead. In order for the delay monitor to work as intended, the minimum delay of the data path must be at least T. Otherwise, outputs from the preceding logic could propagate to the error detecting flop too early, resulting in a false positive detection. In most cases, this should not a difficult constraint to meet, as it is generally critical paths that need to be monitored.

Although this design is simple and efficient, it is not without faults. Using a delayed clock results in significant additional power consumption due to the high capacitance of clock lines. The main problem, however, is that the data path could potentially become metastable, especially when considering the progressive increases in delay associated with transistor degradation. This is due to the fact that excessive delay is detected by the delay monitor only after potentially erroneous data is sampled by the data path flop; the delay monitor will not detect an error until delay T after the rising clock edge, resulting in possible setup and hold time violations on the data path flip flop. A possible solution entails additionally delaying the clock to the data path flip-flop, which is a kind of hard-wired, localized clock stretching. This would reduce timing slack in the next stage of the data path pipeline, which may or may not be allowable depending on the design.

4.1.4 Time-Borrowing Designs

Time-borrowing designs use a latch in place of a flip-flop as the second sequential element. By restricting valid signal transitions to the opaque phase of the latch, the transparent phase of the latch can be utilized as the effective error transition window. In this case, a transition detector must be used in conjunction with the output of the latch. A dynamic logic-based transition detector is presented in work by Das *et. al* [11] and Bowman *et. al* [46]. The design is simple, does not impose a requirement on the timing path, and has low overhead. However, the proposed transition detector requires the design and use of a custom cell and potentially a more complicated instruction replay system [46].

4.1.5 Implementation Flow

Although the time-borrowing delay monitor design seems to be the optimal choice, the data-delay based delay monitor was selected for the prototype DMU implementation due to its simplicity. The prototypical implementation flow is graphically depicted in Figure 4.7. The design RTL is first fed into Design Compiler as normal for synthesis. Afterwards, the synthesis script invokes the inSense library DMU function on the desired design instances. This function scans the selected instances in order to identify the true worst-case path(s) and inserts the necessary delay monitor flip-flops and comparison circuitry. Finally, the DMU insertion method uses Design Compiler to insert delay chain buffers pursuant to the target error window.

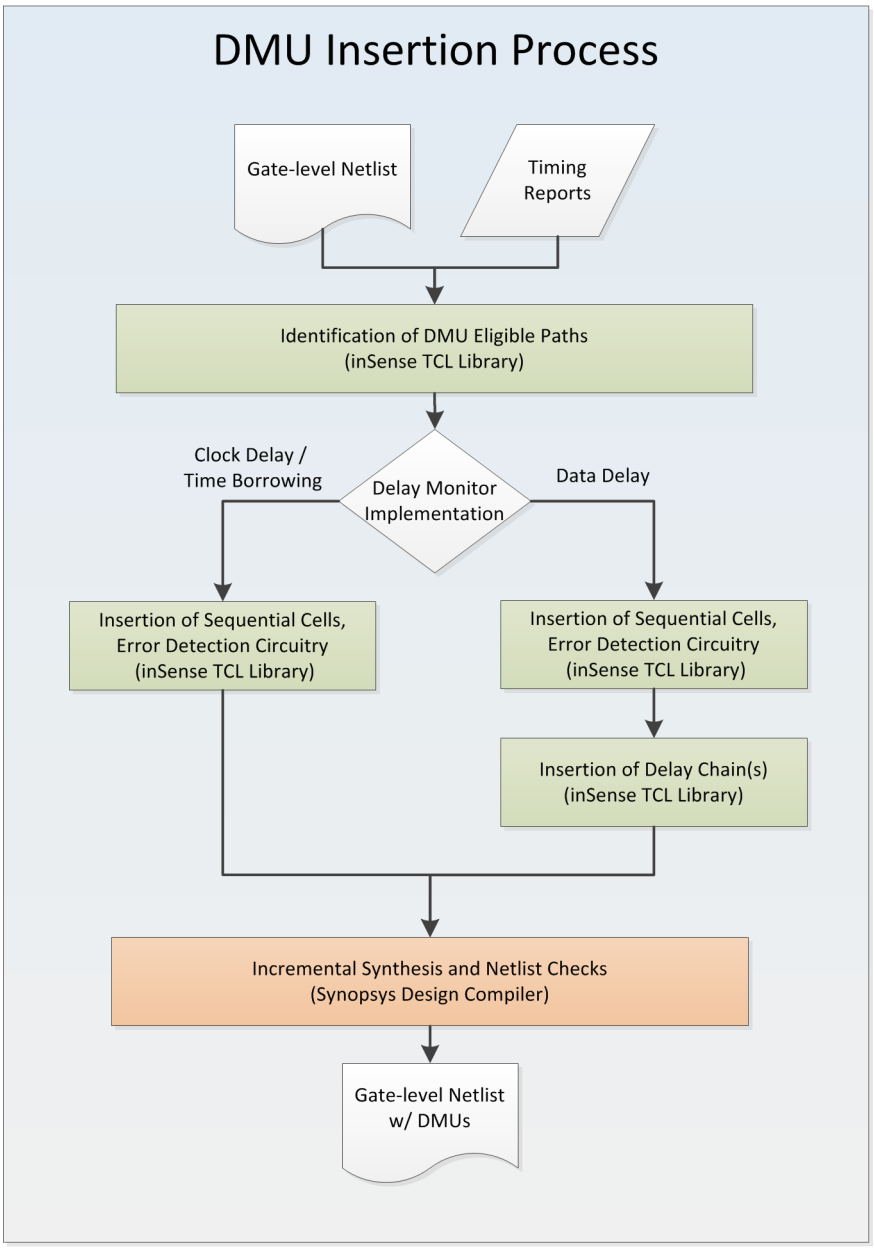


Figure 4.7: DMU Insertion Process

4.2 Variation Control Unit

The variation control unit utilizes data collected by the DMU in a compensatory fashion, allowing the circuit to continue to perform as designed in the presence of process variations, supply voltage and environmental variations, and transistor degradation. Implementation strategies include Adaptive Body Biasing (ABB), Dynamic Voltage and Frequency Scaling (DVFS), or a combination of the two. Although implementation and characterization of a VCU prototype is beyond the scope of this effort, further details of the implementation candidates are examined in the following sections.

4.2.1 Adaptive Body Biasing

An ABB architecture possibly allows for constant circuit delay, enabling full circuit performance under adverse conditions. If each major block within a target design is assigned a dedicated ABB network, intradie process variations could be effectively mitigated while simultaneously minimizing leakage power consumption in other circuit blocks. Previous research has shown an island-ABB architecture to contribute approximately 2-4% additional circuitry and interconnected overhead [4, 34]. Figure 4.8 depicts the block/island-based ABB implementation as is discussed in [14]. Partitioned circuit blocks tie into a central bias generator, which supplies voltages for the PMOS and NMOS sections of each block; this arrangement is compatible with the delay monitoring unit, and the bias generator must include simple logic in order to decode delay detection signals into increased biased voltages for a given circuit partition.

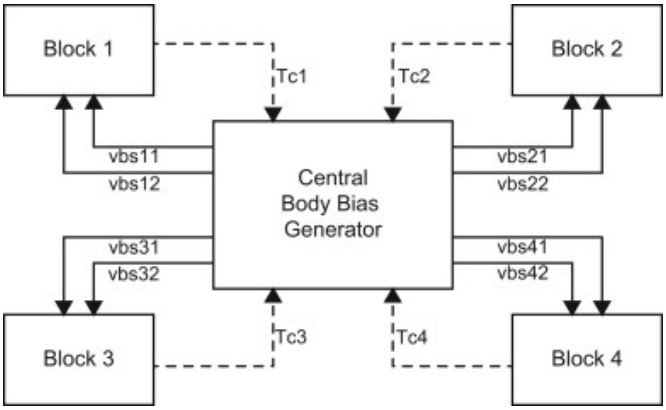


Figure 4.8: Block-Based ABB Implementation Illustration [14]

4.2.2 Dynamic Voltage and Frequency Scaling

DVFS, on the other hand, enables operation of the circuit at lower speeds and voltage levels; its use would facilitate the implementation of a graceful failure mode in the case of excessive process variations and ongoing circuit degradation. DVFS circuitry is already present in most microprocessors and could be tapped by the inSense VCU logic. By combining ABB and DVFS, full system performance could be maintained for certain levels of variation and degradation, with DVFS being enabled to allow for graceful degradation at end of life or for frequency-binned parts.

4.3 Soft Error Detection Unit

In order to mitigate soft errors, each inSense Exoskeleton contains an SDU. The SDU, whether an actual physical unit or implemented through more abstract means, utilizes some type of redundancy in order to detect and potentially correct errors. Aside from attempting process-level hardening, soft errors can be mitigated at the circuit level via device duplication and resizing or at the micro-architectural level through techniques such as redundant multi-threading [42].

The inSense architecture specifies the SDU as a distinct hardware unit for applicability to ASICs and other digital ICs in addition to microprocessors. The inSense ECAD library automatically generates the SDU based on user parameters, minimizing additional complexity in the design process. Upon detection of a transient, the SDU asserts an error signal, requiring the circuit to re-execute the last instruction if the SDU does not contain error correction circuitry. For implementation targeting a microprocessor, this would presumably tie in to the existing pipeline flush logic that is usually active after a branch misprediction or hazard event.

4.3.1 Partial Gate Duplication

In a given circuit, some nodes, when struck by an energetic particle, are much more likely than others to cause an error to propagate to an output [8]. Consequently, these nodes can be duplicated in order to maximize SET detection probability while minimizing area and power overheads. Optimal application of this technique requires a detailed circuit-level analysis in order to accurately rank nodes by their SET-output error probability. While SPICE simulations would yield the most accurate analysis, SPICE simulation is well-known to be very time-consuming and is most likely intractable for large designs. Algorithms which seek to more efficiently model the effect of logical and electrical masking [47, 48] using binary and algebraic decision diagrams provide excellent accuracy with much faster execution times.

4.3.2 Gate Resizing

Once an SET gate-susceptibility analysis has been completed, individual gates can be hardened by increasing their size. Since standard cell libraries generally contain many different sizes of each logic cell, target cells can be replaced with larger sized versions. The efficiency of this technique in terms of area overhead has been demonstrated: Zimanova *et. al.* have reported results of 25% SER improvement for 0.5% area overhead and 67% SER

reduction corresponding to 17% area overhead for SET durations of 50 ps[47]. The gate resizing technique seems to be an efficient method for automatically hardening susceptible nodes. However, available gate sizes are limited to the different-sized cells provided in the target standard cell library, which are meant to provide different drive strengths for varying loads. It could prove difficult to balance cell-resizing with the performance requirements of the system; larger cells may impose a significant impact on design timing.

4.3.3 Implementation Flow

Figure 4.9 details the SDU implementation flow. Although gate/transistor resizing has the potential for superior results, the SDU unit will be implemented at circuit-level using dual modular redundancy for prototyping purposes. The partial gate duplication algorithm [8] is a greedy duplication algorithm which operates on a supplied area constraint. A carefully populated priority queue, ordered by gate SET-susceptibility, is used to select candidates for duplication.

The priority queue is first populated with gates connected to output registers, and as each gate is removed from the queue for duplication, its fan-in gates are added to the queue. This method results in the duplication of a ”cutset” of the output logic, limiting additional circuit loading to the cutset inputs. Gates are dequeued and duplicated until the target area overhead has been met or exceeded, and the final netlist is written out. Algorithms [8, 47, 48] that rely on SPICE simulation of typical workloads or BDD-based analysis can be utilized in order to determine SET-susceptibility prior to queue-insertion. In this work, the gate SET-susceptibility evaluation algorithm utilizes only a weighted combination of the number of flip-flops connected to a gate, the logic distance from an output, and the gate fan-in.

This algorithm is implemented as a function in the inSense TCL ECAD library. The synthesis script invokes the “InsertInsenseSDU” method and an area-overhead target. The library uses Design Compiler to traverse the netlist and duplicate gates in the manner

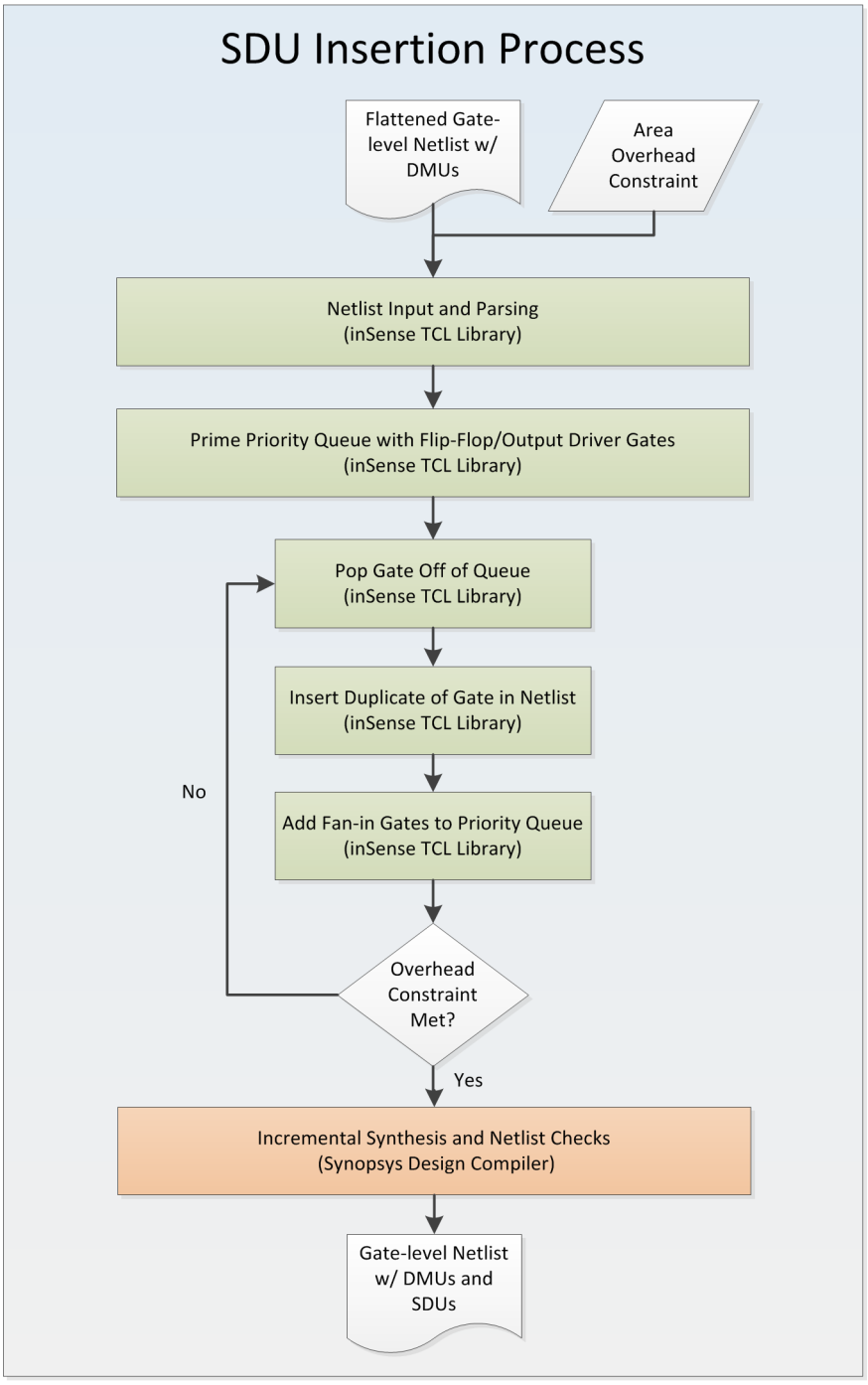


Figure 4.9: SDU Insertion Process

previously detailed. Output flip-flops are duplicated as needed, and after either the area overhead allocation has been exhausted or all gates have been copied, a final incremental compile (allowing only for resizing of cells) is performed in order to reduce or eliminate any potential impact on design timing.

4.4 inSense Design Strategy

In essence, there are two parameters that the inSense architecture is meant to improve: f_{max}, or the maximum frequency at which a design can operate for a given yield target, and transient error coverage. Transient error coverage is limited by the redundancy/area trade-off, while improvement in the former parameter is limited by the efficacy of the combined efforts of the DMU and VCU.

The constraints planning flow representing these considerations is illustrated in Figure 4.10. The upper limit on the improved f_{max} point is constrained by the VCU; in the case where the VCU employs ABB, the maximum delay decrease via forward body biasing determines the new upper limit. If a compensation window is desired for future dynamic mechanisms such as transistor degradations and environmental variations, $f_{max_{improved}}$ would be proportionally reduced. Finally, the new target f_{max} would need to account for the DMU error detection window, which may require some amount of timing slack depending on its implementation.

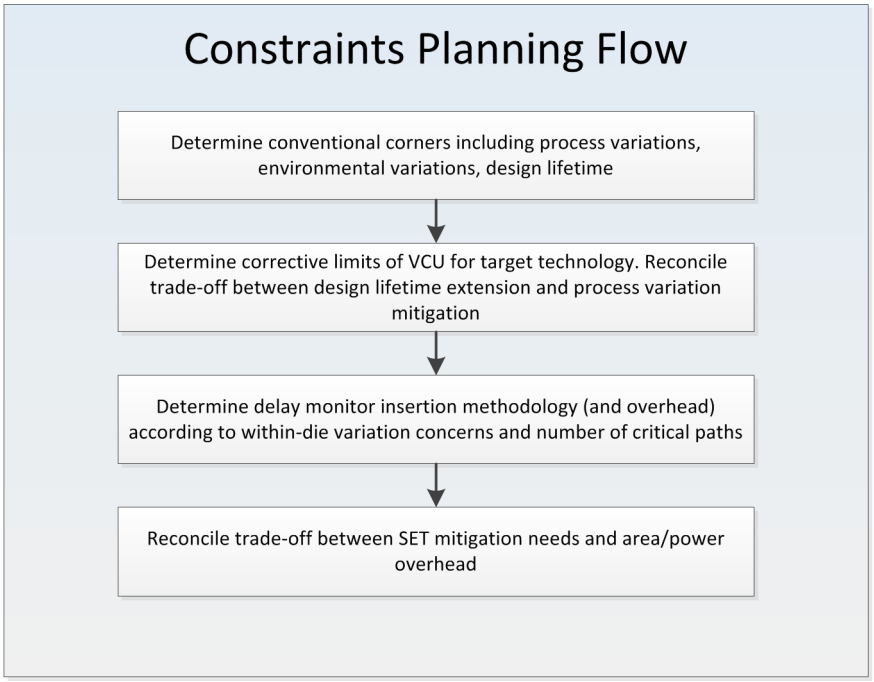


Figure 4.10: inSense Constraints Planning Flow

4.4.1 inSense Calibration Procedure

One consideration involves the effect of within-die process variations and soft errors on the inSense components themselves. The DMU is somewhat affected by local process variations as, by definition, the DMU delay chain is the critical path. Generally, the effective error detection window will either shrink or lengthen for fast or slow corners, respectively. Extreme local process variations might result in the DMU could becoming non-functional. A post-fabrication calibration method, however, could serve to alleviate these concerns:

1. Increase forward body bias and/or slow clock speed to meet inSense worst-case frequency target
2. Execute test instructions and verify execution (i.e. assert that the error signal is asserted)

3. Decrease forward body bias and/or increase clock frequency until point of first error detection

4. Increase forward body-bias and/or decrease clock frequency one step for nominal operation at maximum delay detection window.

This calibration method could also include a provision for reverse body biasing on designs and logic blocks with fast-corner process variations. The SDU is negatively impacted by large local process variations, as timing violations could cause failures. In cases where the DMU and SDU are common at the block level, it is likely that the two components would see similar WID process variation effects; consequently, calibration of the DMU would also be applicable to the SDU.

The DMU is not particularly sensitive to SETs, as the worst-case effect would be a temporary false-negative error indication. This could result in an error for cases where the error detection window is very small and a missed delay increase is enough to cause an error on the data path, but the combination of these conditions is unlikely. A false-positive indication is of little concern, as the only result would be to cause the VCU to over-compensate for a fallacious delay. With a PID-style control system, the effect of a false-positive would be removed from the system after some duration.

Chapter 5

Testing and Results

The functionality and performance of the proposed architecture were characterized via implementation in ISCAS '85 circuits. Aside from the benefit of using standardized benchmark circuits, the ISCAS '85 units are convenient as they are purely combinatorial (*i.e.* minimal test cases) and span a range of circuit sizes. A description of each design in the benchmark suite is available in Table 5.1. All simulations were run on post-synthesis netlists; this allowed sufficient resolution for testing and characterization while minimizing unnecessary effort not directly applicable to demonstrating the goals of this research.

Circuit	Function	No. of input lines	No. of output lines	No. of logic gates	No. of major functional blocks
c432	27-channel interrupt controller	36	7	160	5
c499	32-bit SEC circuit	41	32	202	2
c880	8-bit ALU	60	26	383	7
c1355	32-bit SEC circuit	41	32	546	2
c1908	16-bit SEC/DED circuit	33	25	880	6
c2670	12-bit ALU and controller	233	140	1,193	7
c3540	8-bit ALU	50	22	1,669	11
c5315	9-bit ALU	178	123	2,307	10
c6288	16×16 multiplier	32	32	2,406	240
c7552	32-bit adder/comparator	207	108	3,512	8

Table 5.1: ISCAS '85 Benchmark Circuit Listing[1]

The test methodology defines functional verification testing via application of test vectors to the post-synthesis circuits and their RTL-level models followed by successful comparison of the generated outputs. Automatically generated VHDL test benches, depicted

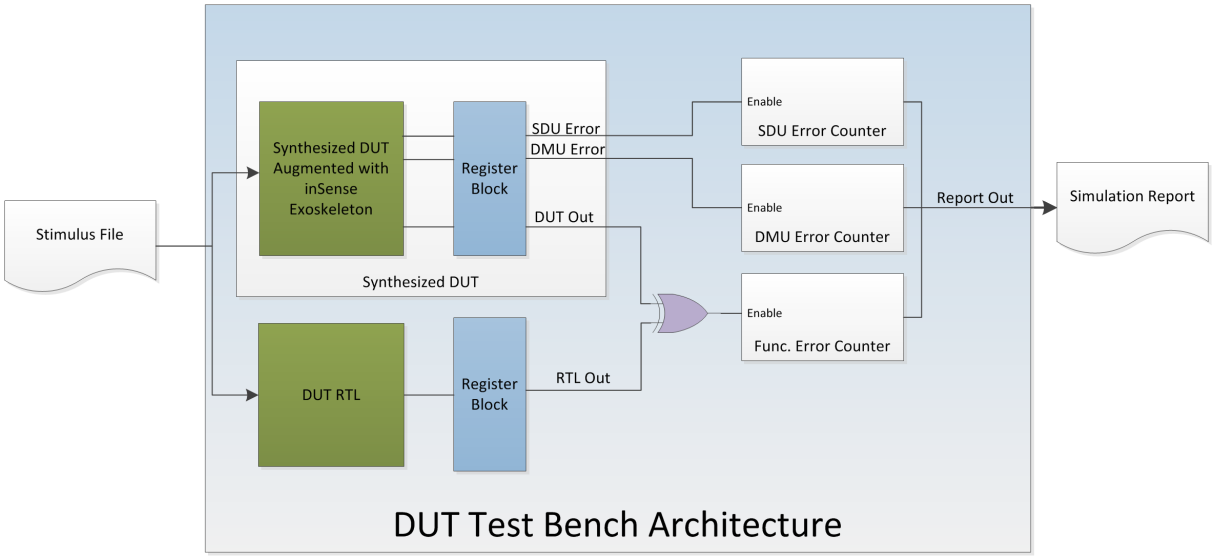


Figure 5.1: Testbench Design

in Figure 5.1, provided this capability. inSense augmented designs were simulated over a sweep of key parameters in order to characterize functionality and performance of the SDU and DMU. The DMU was tested against a span of effective circuit delay values using worst-case vectors generated using Synopsys PrimeTime “true-path” static timing analysis. The SDU was characterized for error coverage vs. area and power overheads. Pseudo-random test vectors were used for the SDU testing, generated via a custom Perl script. In all cases, the target simulation platform was Synopsys VCS MX.

Figure 5.2 details the harness flow by which environment-preparation scripts readied each design for testing: the original ISCAS netlists were converted from “.bench” format into VHDL followed by generation of directory trees for implementation and simulation. Generic scripts and templates including makefiles, implementation scripts, and simulation files were modified to include design-specific information.

Each circuit was implemented using both the 90nm and 32nm versions of the Synopsys Armenian Education Department (SAED) process design kits (PDKs) [49, 50]. These PDKs mimic commercially-supplied packages and are open and accessible at each level

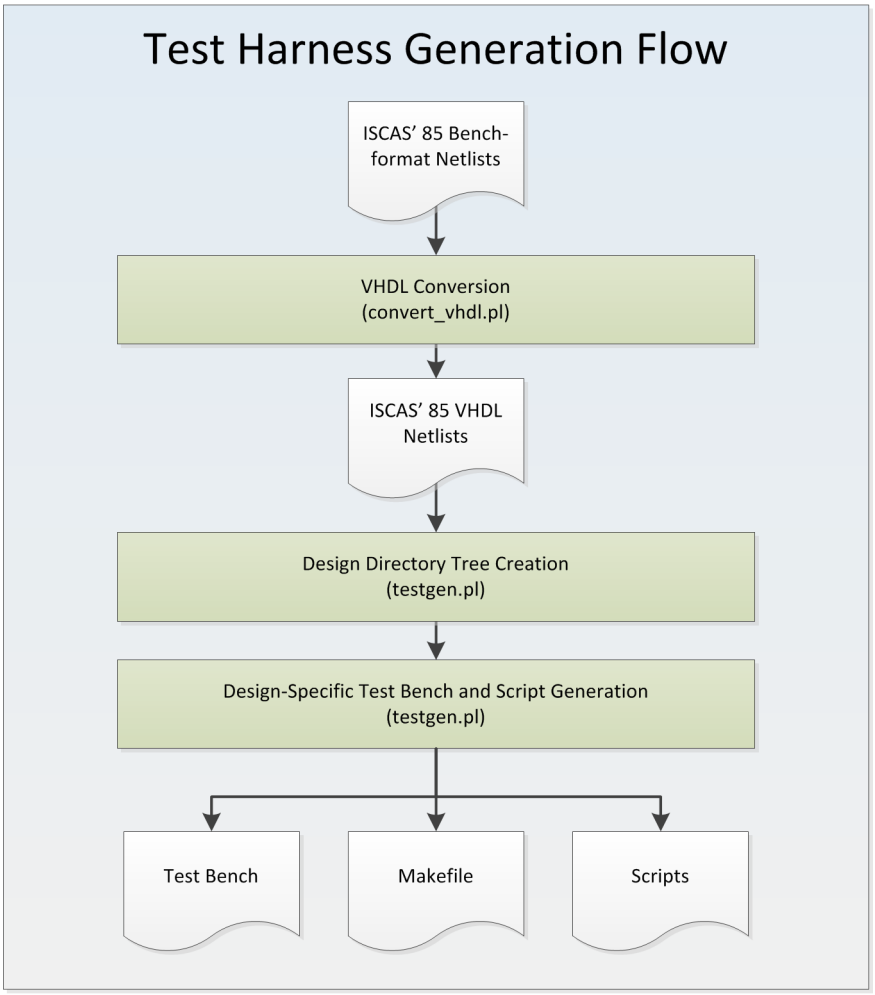


Figure 5.2: inSense Test Harness Generation

Library Content	Description
Technology Kit	Schematic symbols, Verilog and VHDL simulation models, DRC and LVS decks, HSPICE netlists, extracted C/RC netlists, technology files
Digital Standard Cell Library	>340 (SAED90nm) / 350 (SAED32nm) combinatorial and sequential logic cells of different sizes/drive strengths, low-leakage and high-speed cells for multi-V_T design paradigm
I/O Cell Library	Wire-bond and flip-chip versions of library, including digital and analog I/Os, power and ground pads for different loads
Memories	Dual-port medium-sized SRAMs
Phased Lock Loop	3-mode PLL Circuit
References	Reference designs (OpenSparc/PowerPC for SAED90nm, Orca/ChipTop for SAED32nm), megacells, examples

Table 5.2: SAED90nm/SAED32nm PDK Contents with Descriptions

of the design kit hierarchy, making them especially suitable for VLSI research efforts. Table 5.2 contains a listing of the contents and capabilities of these PDKs. For this analysis, the standard cell library with typical PVT conditions for each of the digital standard cell libraries were employed; in particular, the "saed90nm_typ" library was used for the 90nm PDK, and the "saed32hvt_tt1p05v25c" library was used for the 32nm PDK.

Table 5.3 lists each of the ISCAS '85 benchmark circuits along with their target clock periods and the associated areas achieved when using each PDK. Upon examining this table, an oddity becomes evident: it appears that the circuits generated with the 32nm version of the library are in fact larger than those synthesized using the 90nm version of the library. This is attributable to an error in the included SAED32_hvt standard cell library binary database - the area values for each cell in the library do not match the documentation and in fact are identical to those of the SAED90nm library. The following sections further detail the test methodologies for the DMU (5.1) and the SDU (5.2).

Design	Clock Period (ps)		Area (μm^2)	
	SAED90nm	**SAED32nm**	**SAED90nm**	**SAED32nm**
c432	1850	850	1246.528	1924.006
	1950	900	1294.577	1805.154
	2050	950	1265.355	1667.402
c499	1400	870	2815.618	2714.093
	1500	920	2839.066	2714.093
	1600	970	2839.066	2714.093
c880	1700	870	2351.678	2749.871
	1800	920	2351.678	2870.648
	1900	970	2351.678	2872.777
c1355	1450	780	2707.566	3021.986
	1550	830	2707.566	3173.300
	1650	880	2707.566	3011.275
c1908	2050	1100	2691.602	2978.830
	2150	1150	2980.234	2981.788
	2250	1200	2934.621	2990.680
c2670	1500	710	7194.151	8012.405
	1600	760	7336.146	7907.933
	1700	810	7493.106	7751.780
c3540	2770	1370	5045.455	6022.803
	2870	1420	5016.122	6259.502
	2970	1470	5074.709	6029.738
c5315	1650	870	9487.756	10854.487
	1750	920	9669.983	10668.420
	1850	970	9474.661	10826.763
c6288	5000	2630	12605.675	23467.428
	5100	2680	12337.184	24269.317
	5200	2730	12932.143	22805.901
c7522	1800	1550	11993.602	10772.688
	1900	1600	11663.068	10772.688
	2000	1650	11890.984	10772.688

Table 5.3: ISCAS '85 Circuits: Target Clock Periods for Synthesis and Corresponding Areas, SAED PDKs

Constraint	Value (90nm)	Value (32nm)	Notes
Library	saed90nm_typ	saed32hvt_tt1p05v25c	Typical corner operating conditions. Standard threshold voltage library is used with saed90nm, and high voltage threshold library is used with saed32nm
Input/Output Delay	0 ns	0 ns	Constraint is not meaningful for these tests. Removed to simplify results analysis.
Input drive strength	1 kΩ	1 kΩ	Output impedance driving input pins of design
Output load capacitance	7.77 fF	2.6 fF	Output load is 5 times the library-defined load of a standard-drive D flip-flop
Wireload Model	Design-size dependent	Design-size dependent	Automatically changes depending on design size as defined by library

Table 5.4: Constraints used for synthesis (DC Compiler) and Timing Analysis (PrimeTime)

5.1 Delay Monitoring Unit

Variable-sized error windows in the DMU were implemented by synthesizing each circuit with a fixed delay target, varying the clock speed to create different amounts of slack, and allowing the DMU implementation script to fill the remaining slack via the delay monitor chain. DMU functional verification and performance testing was accomplished via simulations with successive increases in input vector delay and correlating the number of detected errors per delay value, the DUT area, and the power consumption of the DUT; these data were analyzed in order to characterize the delay-detection error windows.

Figure 5.4 presents the testing methodology for the DMU, and shmoo-style plots characterizing the DMU over both the SAED32nm and SAED90nm PDKs are included in Figures 5.5 and 5.6. Shmoo plots (see Figure 5.3) have been used in industry to illustrate

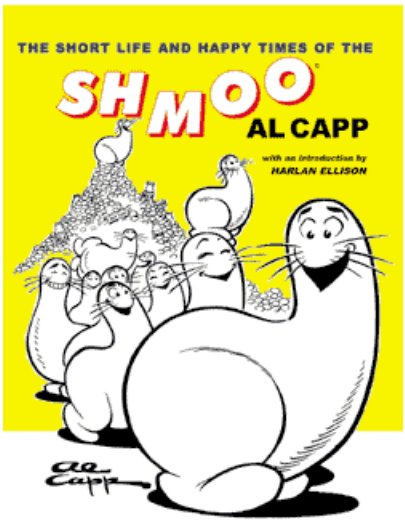


Figure 5.3: The Shmoo, by Al Capp

memory performance over varying supply voltage and operating frequency conditions and are so-named due to the shape of plots generated from testing magnetic core memory arrays. well suited to displaying the DMU testing results.

Shmoo plots are presented for each of the ISCAS'85 benchmark circuits, each of which contains discrete points of one of four different colors: white, yellow, green and red. The white section illustrates an error-free data path computation with no error detected by the DMU. The yellow points denote an error-free data path computation where the DMU may or may not flag an error, as the setup/hold times for the DMU flip-flop at the end of the delay chain have been violated. The green regime illustrates the reliable error detection window, wherein there is no error on the data path, but the DMU error signal is asserted as an early warning. The red region indicates a data path error.

The shmoo plots reveal that it is somewhat difficult to obtain completely consistent error window characteristics across multiple designs for a target window size. One limitation involves the minimum delay buffer available in the employed standard cell library, which determines the effective step size in error window length adjustments. The minimum-delay buffer in the SAED32nm_hvt and SAED90nm standard cell libraries, "NBUFFX2", has an

average in-circuit propagation delay of 44 picoseconds for the former library and 77 ps for the latter. This corresponds to the observed window step size of $\approx$ 50ps in the SAED32nm shmoo plots and $\approx$ 100ps in the SAED90nm shmoo plots. Secondly, the amount of slack (white area) that is filled by the error-detection window depends on the original (pre-DMU) slack of the target output, the driving capability of the cell at that output, and the combined delay characteristics of the library buffer cells that can fill the available slack without resulting in a negative slack on the DMU path.

An additional obstacle is that correlation of the test results versus the implemented DMU delay chain depends on the accuracy of the worst-case vectors provided by Prime-Time; for some designs, it is difficult to get the tool to produce a true worst-case vector. If a design has many long, complex critical paths, such as for benchmark c6288, the tool will not be able to identify the worst path in a reasonable amount of time. Also, some of the "true" PrimeTime paths are false (*i.e.* overconservative) in that PrimeTime can fail to consider the inertial property of gate propagation delay and does not correctly account for the control-to-output delay on multiplexer cells. In the former case, some paths should not exist as a signal with a pulse-width duration less than a gate's propagation delay will not propagate; in the latter case, PrimeTime can miss the fact that the propagation to output delay of a multiplexer does not apply if the signal applied to the mux control switch does not change.

Beyond functional testing, metrics of interest for the data-delay DMU implementation include comparison of area and power overheads for increasing error window sizes. These results are presented in the following subsections.

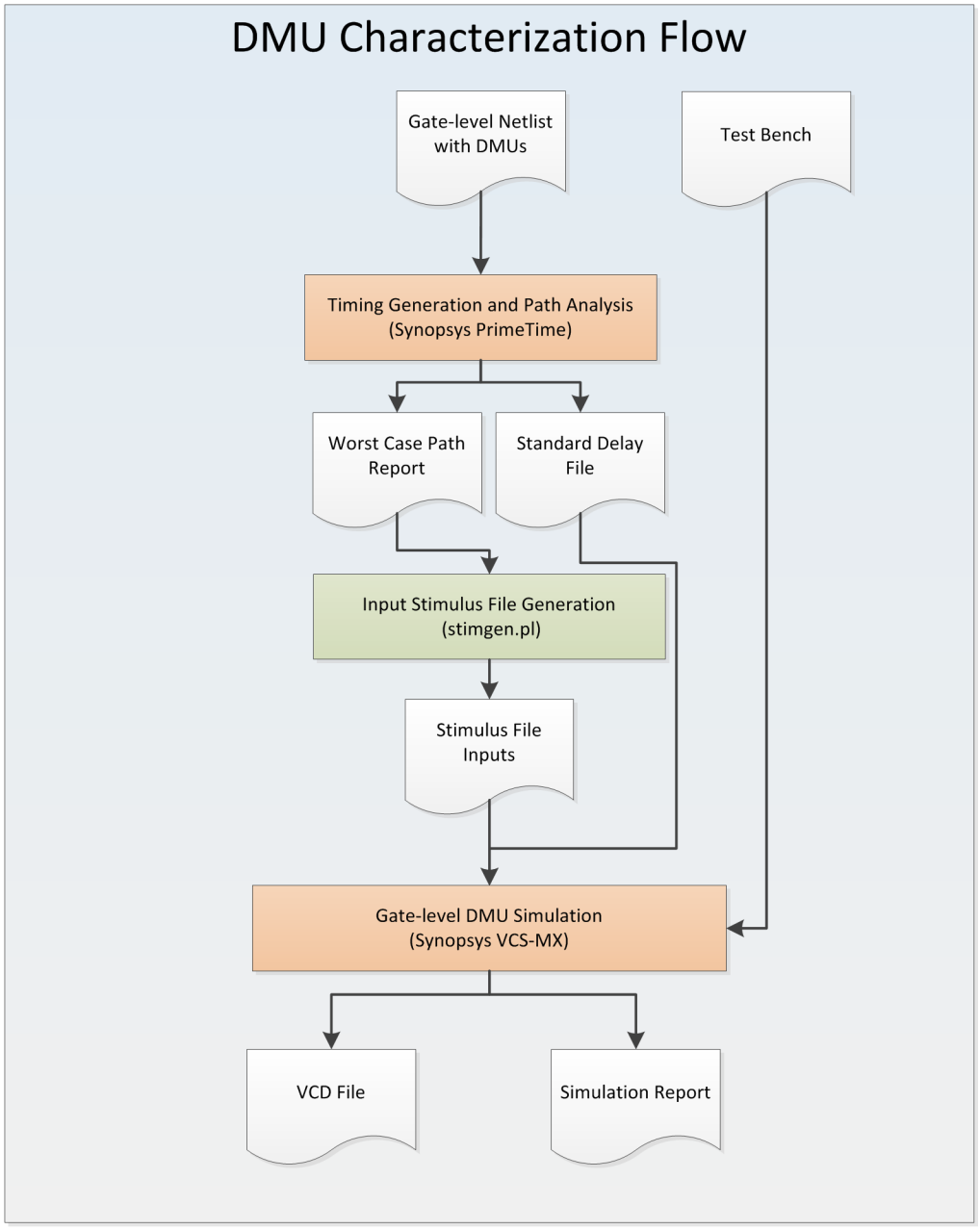
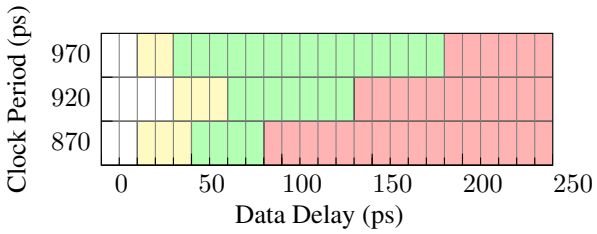
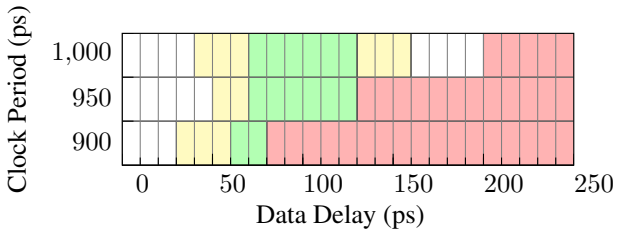


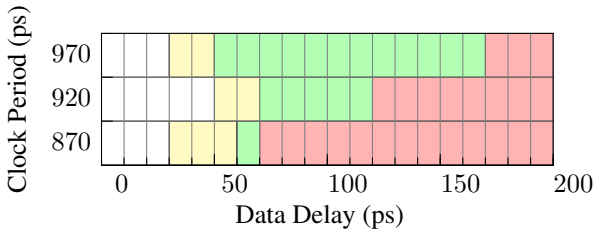
Figure 5.4: DMU Characterization Flow



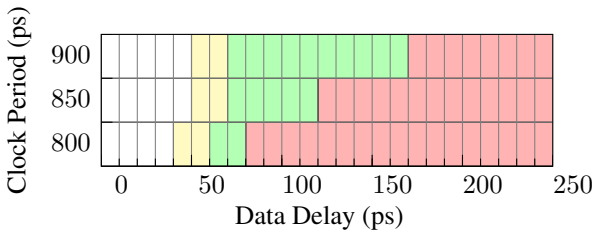
(a) c432



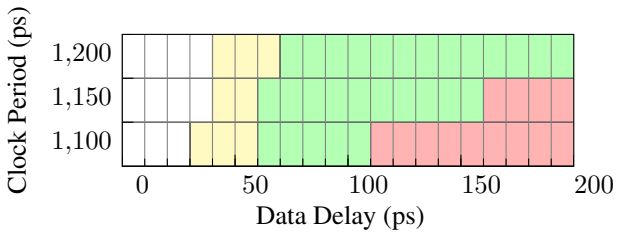
(b) c499



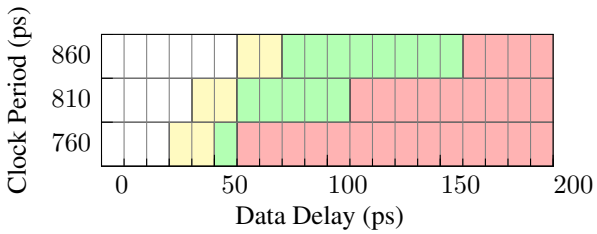
(c) c880



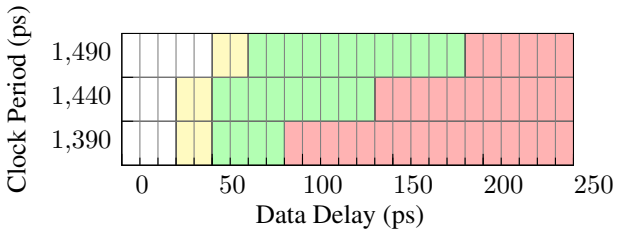
(d) c1355



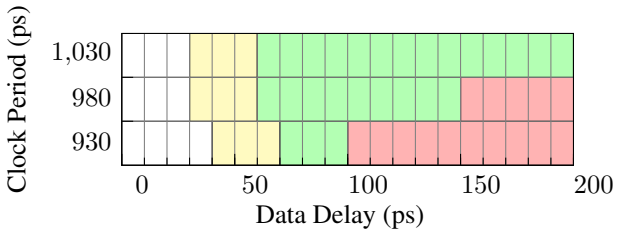
(e) c1908



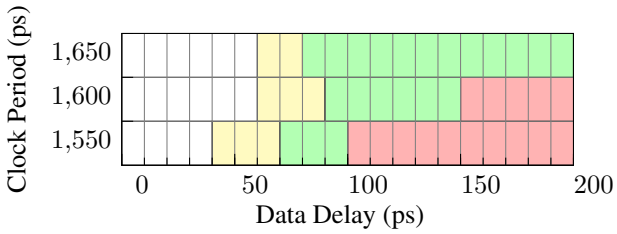
(f) c2670



(g) c3540



(h) c5315



(i) c7552

Figure 5.5: ISCAS ’85 Shmoo-Style Delay Detection Plots, SAED32nm Library

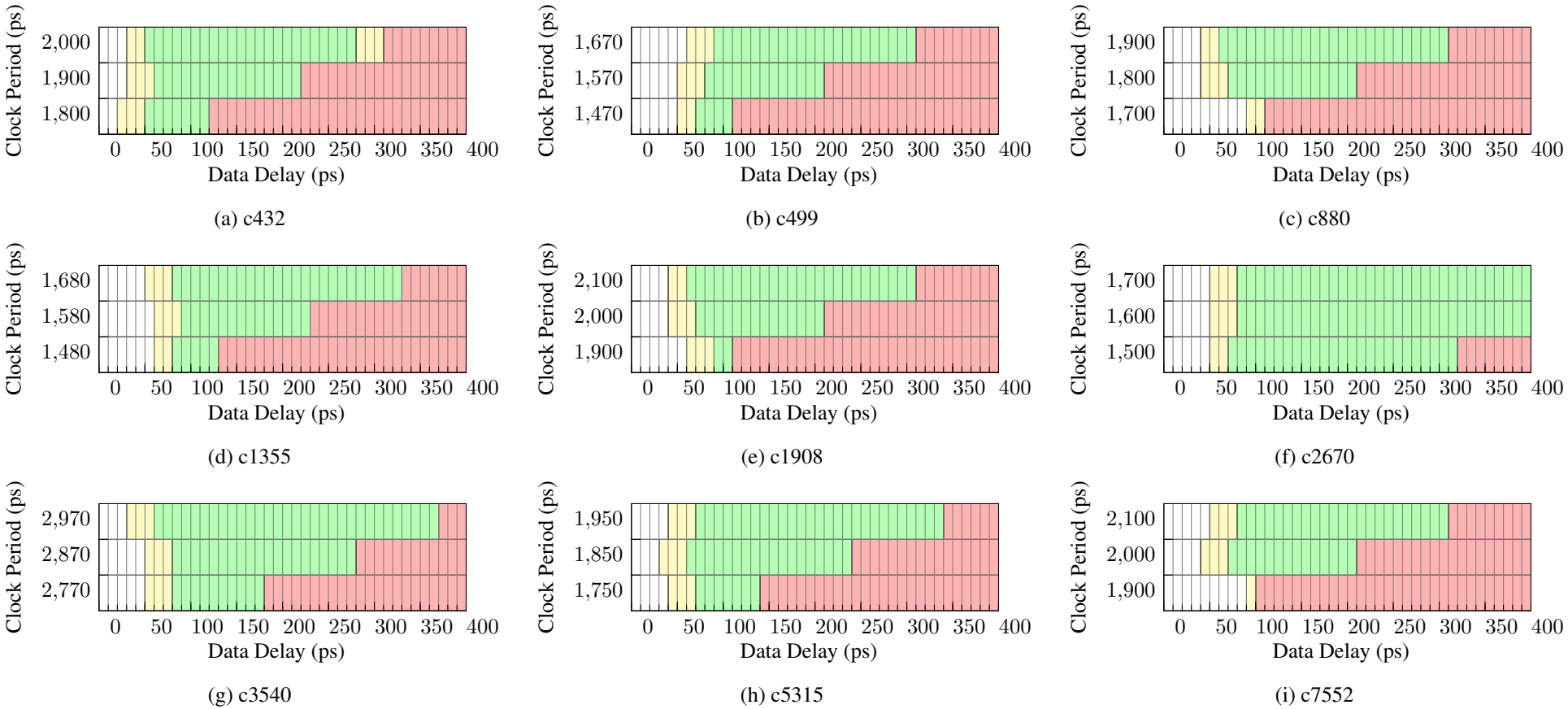


Figure 5.6: ISCAS ’85 Shmoo-Style Delay Detection Plots, SAED90nm Library

5.1.1 Area Overhead vs. Delay Detection Window

Figures 5.7 and 5.8 illustrate the relationship between incurred area overhead and delay detection window for each of the ISCAS '85 benchmark circuits. Besides c499, all circuits incurred 5% or less area overhead for each target delay detection window. Many designs, including c2670, c3540, c5315, and c7552 incurred less than 2% area overhead for each delay detection window.

The non-linear trends for some of the designs are owed to the fact that the DMU insertion process prioritizes the size of the error window over area or power cost. This confirms that it is possible (by design) for the cell overhead to vary non-monotonically from one error window size to the next depending on which combination of buffers comes closest to the desired error window. Take, for example, the second and third points from the c1908 characteristic in Figure 5.7. The DMU insertion process utilized an "NBUFFX32" buffer for the 75ps error window, which has an area of $10.67\mu m^2$, while a "DELLN1X2" cell is used for the 115ps error window and has a significantly smaller footprint of $5.53\mu m^2$.

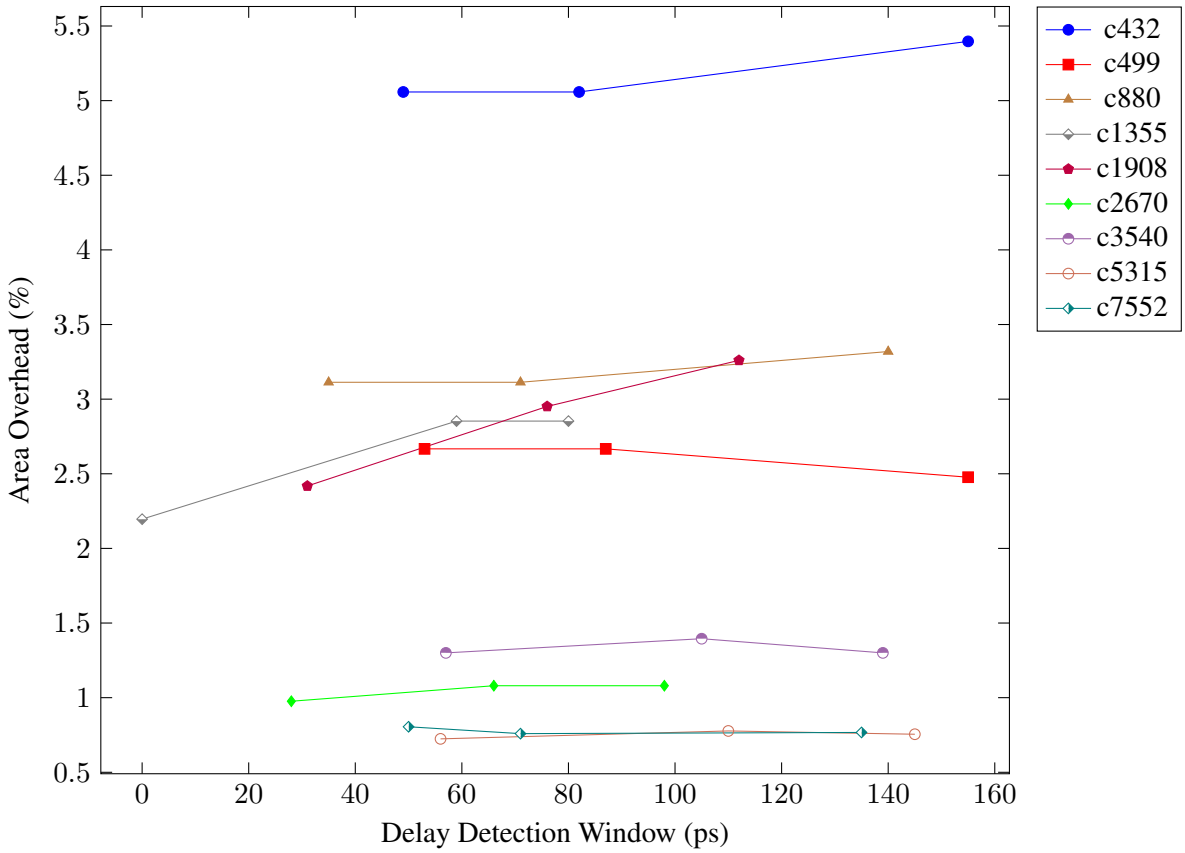


Figure 5.7: Area Overhead vs. Delay Detection Window for inSense DMU, SAED32nm, ISCAS ’85 Circuits

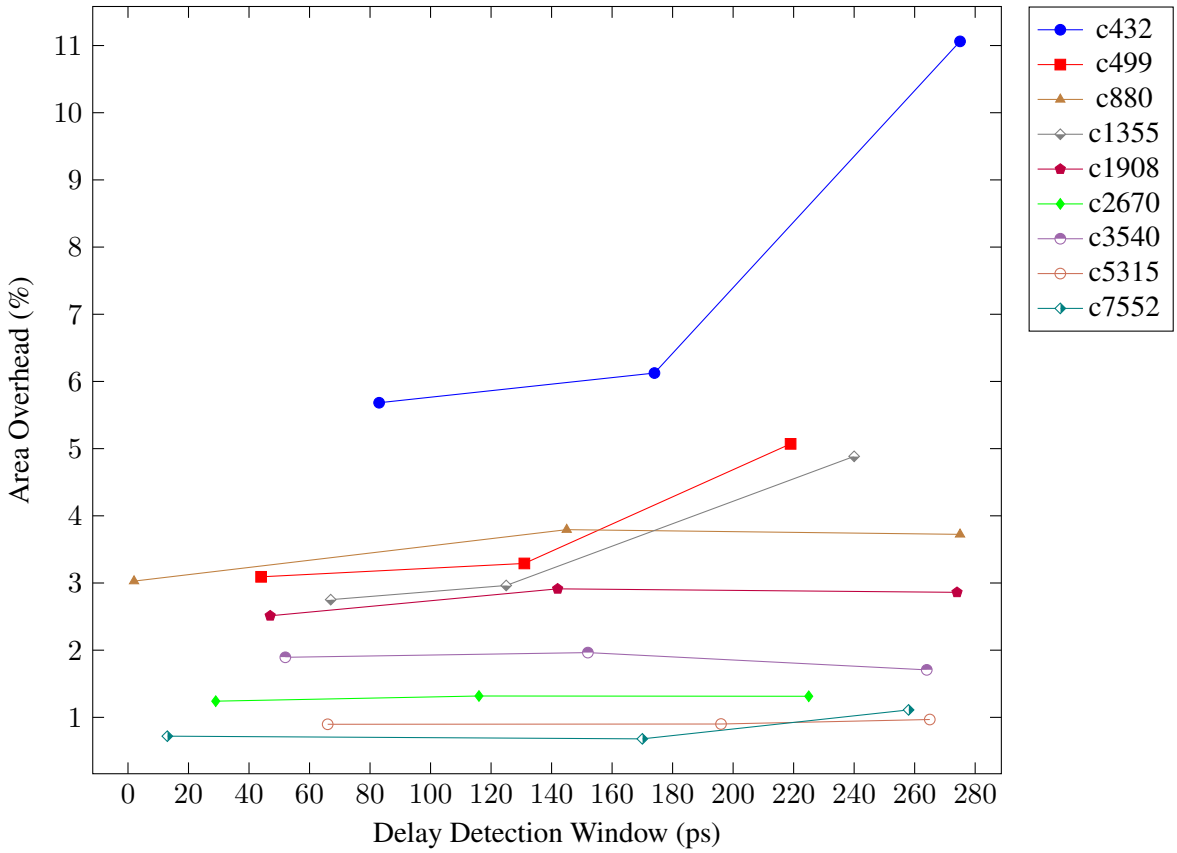


Figure 5.8: Area Overhead vs. Delay Detection Window for inSense DMU, SAED90nm, ISCAS '85 Circuits

5.1.2 Power Overhead vs. Delay Detection Window

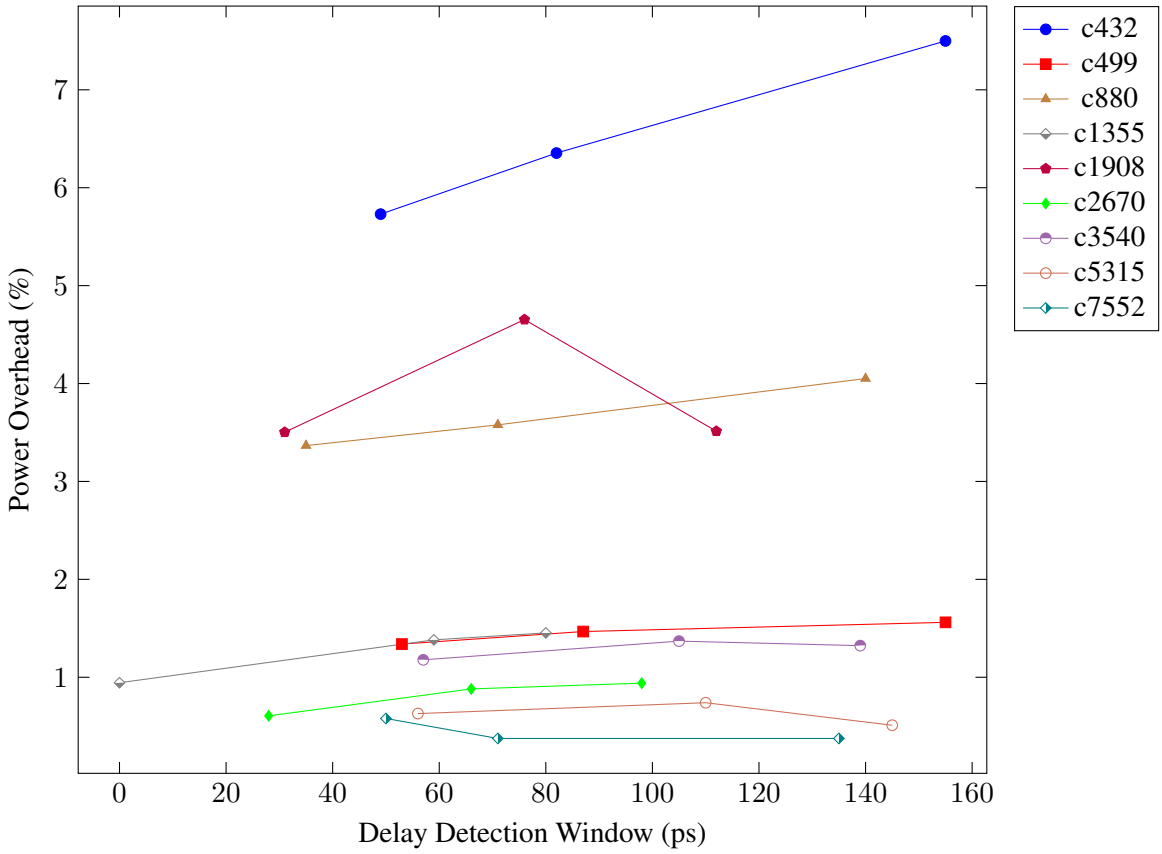


Figure 5.9: Power Overhead vs. Delay Detection Window for inSense DMU, SAED32nm, ISCAS '85 Circuits

Figures 5.9 and 5.10 illustrate the relationship between estimated power overhead and error detection window for each of the ISCAS '85 benchmark circuits. As is expected, the power overhead results are commensurate with the area overhead results. Larger circuits, in general, achieve better results. The power (and area) anomaly for the C499 circuit are explained by the same reason supplied in Section 5.1.1. Examining the circumstances of the third data point in Figure 5.10 for c499, the DMU insertion process selected cell "IBUFFX32", which is a large, power-hungry cell. It consumes approximately 56μW of

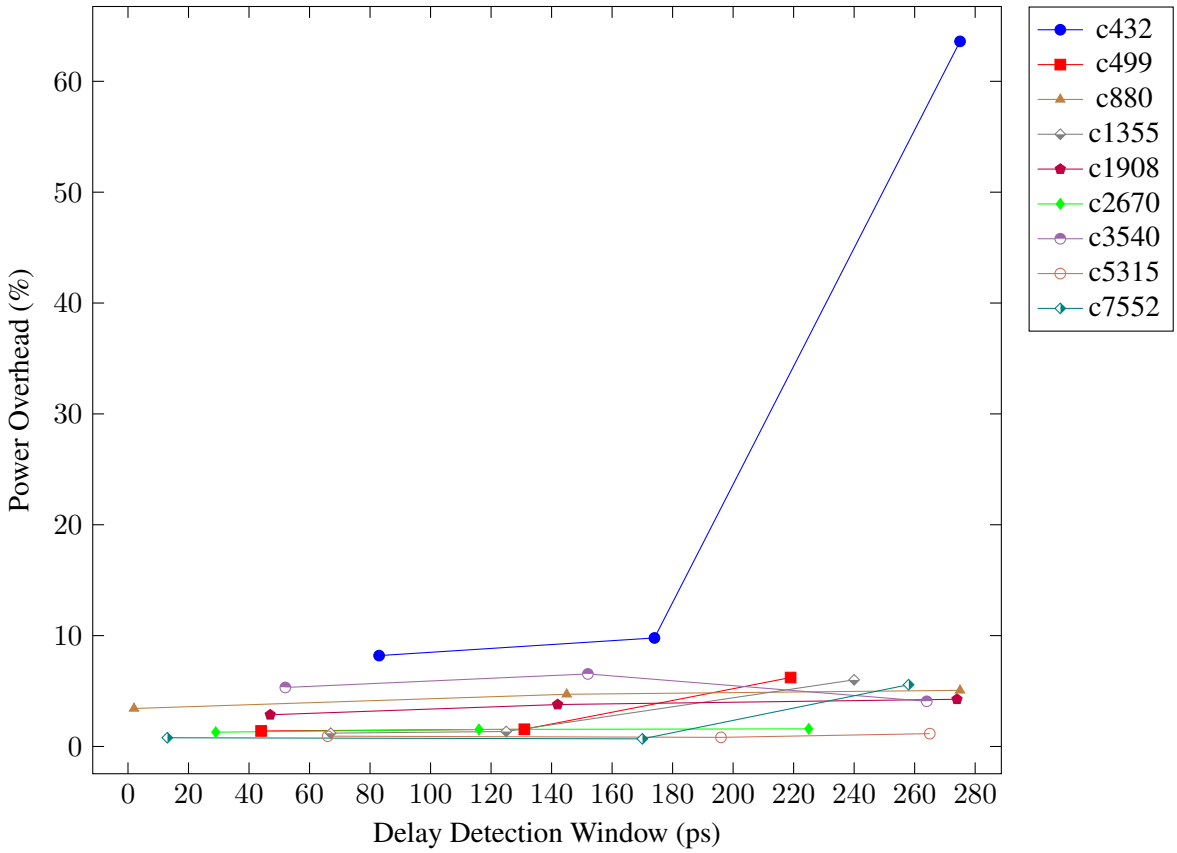


Figure 5.10: Power Overhead vs. Delay Detection Window for inSense DMU, SAED90nm, ISCAS '85 Circuits

the 191μW augmented netlist; in comparison, the original netlist power consumption was reported to be 117μW.

5.2 Soft Error Detection Unit

Fundamentally, characterization of the Soft-Error Detection Unit is accomplished through simulation of injected SETs in an inSense-augmented design; the specifics of this process are illustrated as a flow diagram in Figure 5.11. The synthesized netlist is first converted into SPICE using the Synopsys nettran utility, and a custom script ("sp_flatten.pl") flattens the design hierarchy. This allows the SET injection script ("set_injection.pl") to inject SETs into different nets without the possibility of undesired replication with subcircuit instantiations.

Verification of the design is facilitated through the Synopsys VCS-MX/Nanosim co-simulation feature. For each design, VCS-MX runs a custom VHDL testbench and interfaces with Synopsys NanoSim, which simulates the spice model of the inSense augmented design. This configuration, as is illustrated in 5.1, allows for the advanced verification capabilities of VHDL in tandem with low-level device modeling afforded by SPICE simulation. The Synopsys SAED90nm PDK supplies a SPICE library with models for each of the available transistor families as well as characteristic models for each of the supplied standard cells.

In this work, SETs are modeled as a SPICE current source injected into a given node, with the rise and fall times of a typical pulse caused by a low-energy neutron interaction [51]. This is an established simulation method and well-represents the physical phenomenon; in fact, this has been confirmed through measurements by Gill *et. al.* with technology as recent as the 32nm node [27]. The duration of the simulated SETs, however, are lengthened to one full clock-cycle period for two reasons: we are chiefly concerned with error coverage rather than pulse characteristics, and longer pulse widths seem to better represent results for advanced technology nodes. SETs are injected only into combinational

logic in our test methodology, as SDU efficacy is the metric of interest. Testing is accelerated by injecting precisely one SET into a random node during each simulated clock cycle, drastically reducing simulation time requirements.

The SDU insertion process was functionally verified by simulating each design over the span of area overheads (0% - 100%) and correspondingly verifying 0% and 100% error coverage at the boundary points. In addition, each design was verified to indicate no errors over the span of area overhead targets when SET injection was disabled. In order to clearly identify the performance characteristics of the SDU, the simulation vector is psuedorandomly selected for the original flattened SPICE netlist of a target design and re-used for each SDU-augmented netlist for direct comparison. Performance metrics of interest include error coverage versus area and power overhead and are presented in the following sections.

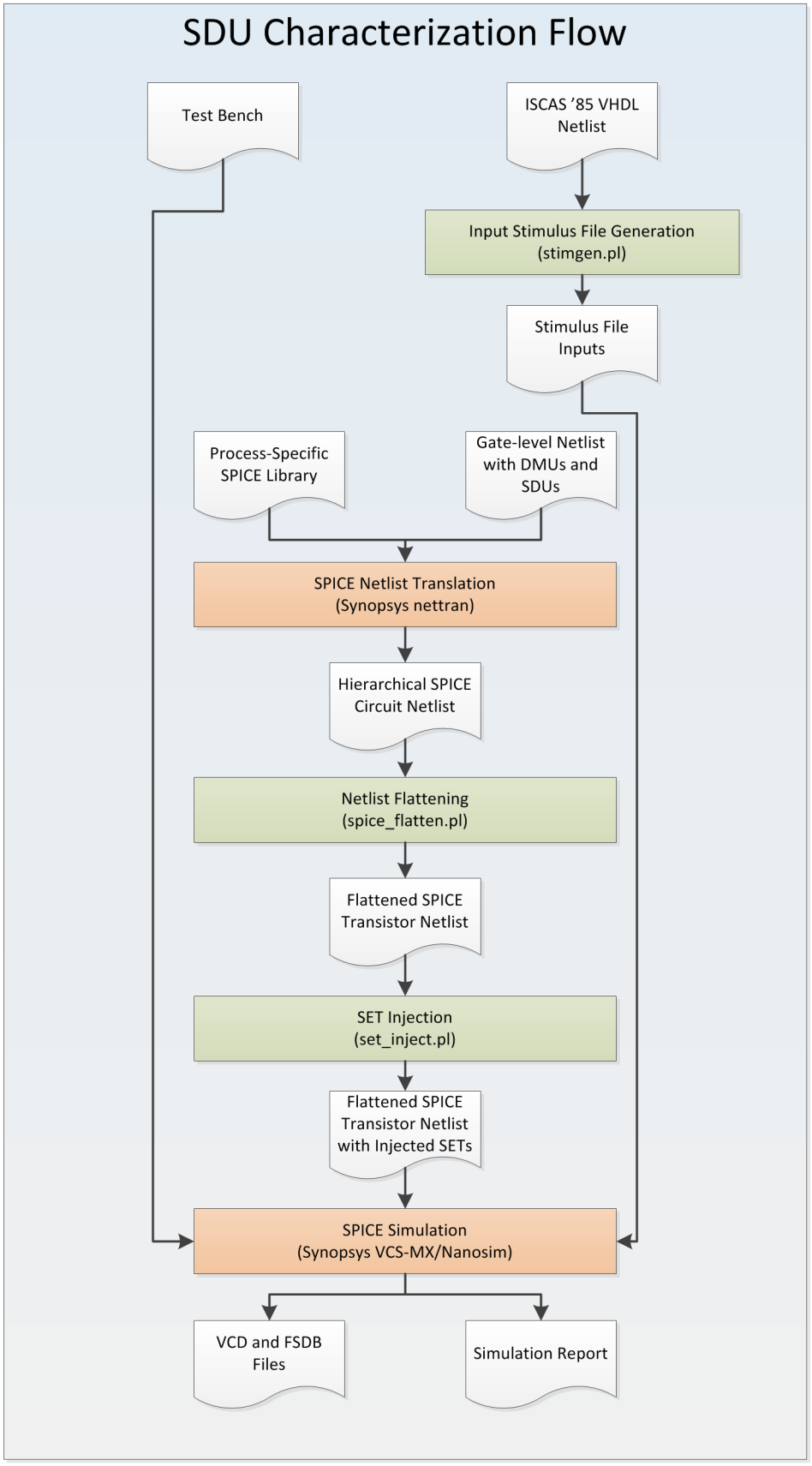


Figure 5.11: SDU Characterization Flow

5.2.1 Error Coverage vs. Area Overhead

Plots depicting SET error coverage vs. area overhead results for the inSense Exoskeleton including both the DMU and SDU are presented in Figures 5.12 and 5.13. As a reference, each plot contains a $y = x$ plot (depicted by a black line without markers) to compare and contrast results against a standard DMR approach. In general, the SDU insertion algorithm appears to provide error coverage which is only slightly better than the amount of incurred area overhead. This is unsurprising given that a naive algorithm is used for the gate duplication; superior results should be attainable with one of the other algorithms reviewed in Section 4.3.

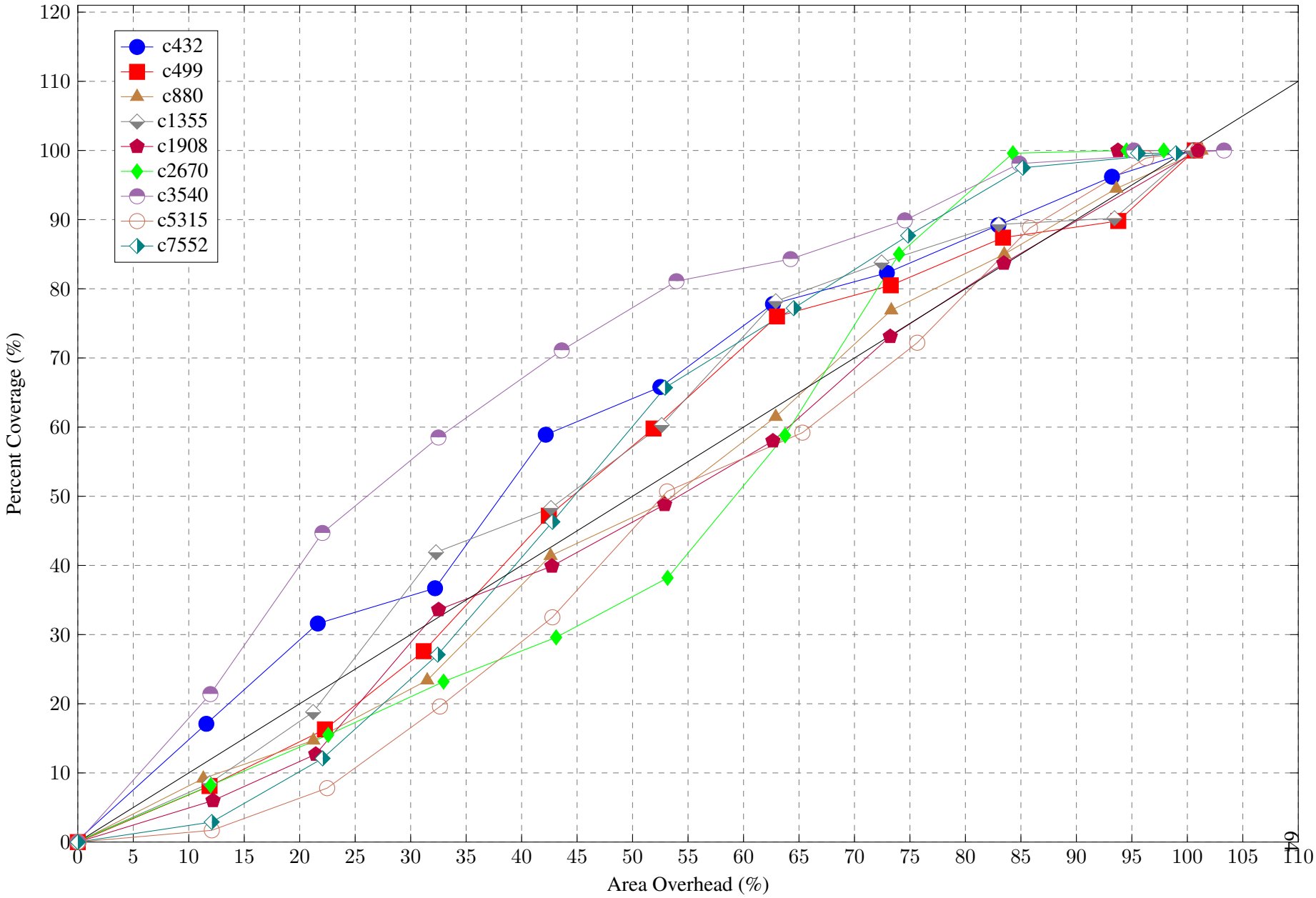


Figure 5.12: inSense Exoskeleton SET Error Detection Rate vs. Area Overhead, ISCAS ’85 Circuits, SAED32nm HVT Library

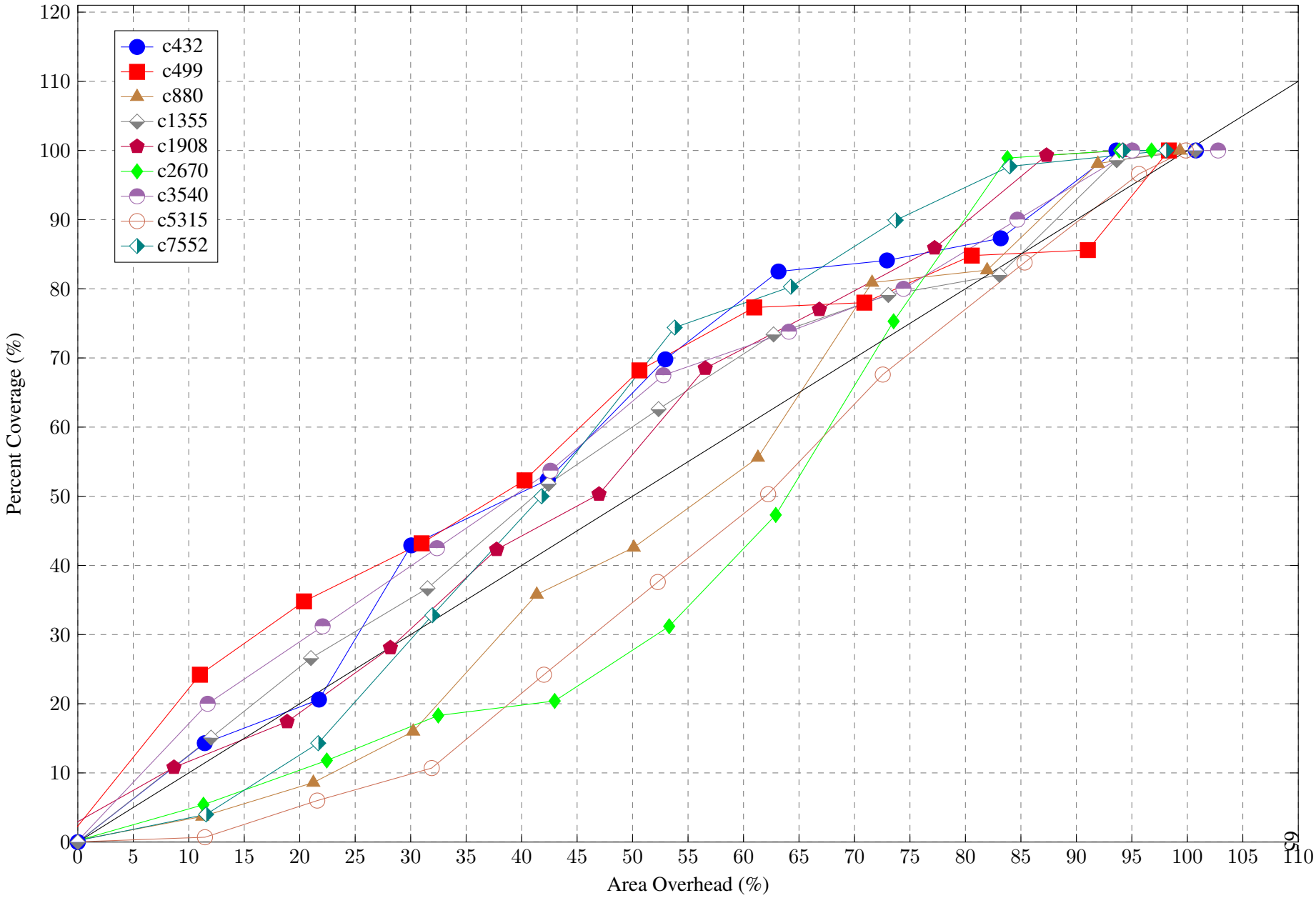


Figure 5.13: inSense Exoskeleton SET Error Detection Rate vs. Area Overhead, ISCAS '85 Circuits, SAED90nm HVT Library

5.2.2 Error Coverage vs. Power Overhead

Plots illustrating SET error coverage vs. power overhead results for the inSense Exoskeleton including both the DMU and SDU are presented in Figures 5.14 and 5.15. Again, designs c2670, c3540, and c5315 appear to benefit from better than average results, while c432, c1355, and c1908 appear to be somewhat worse than average.

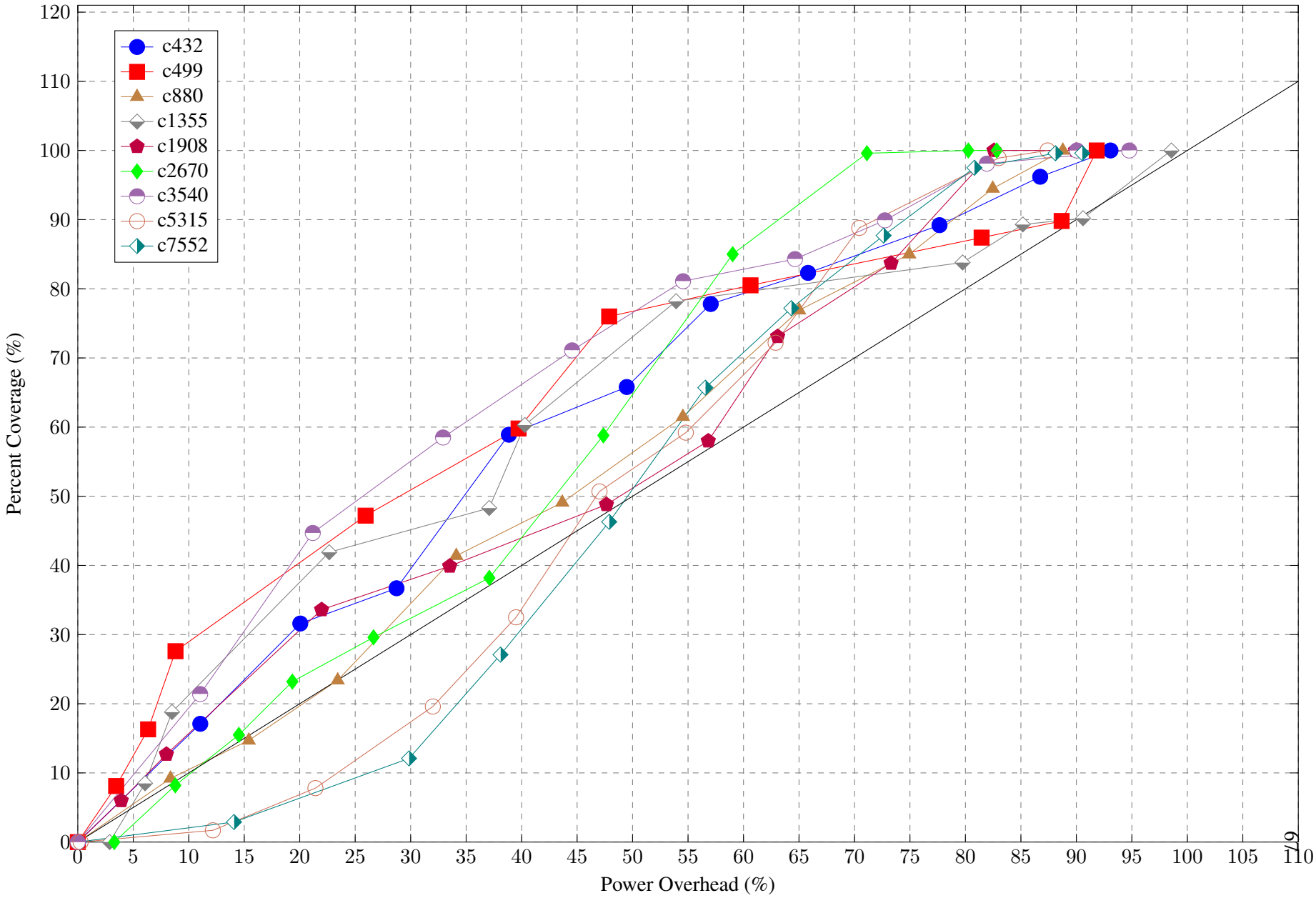


Figure 5.14: inSense Exoskeleton SET Error Detection Rate vs. Power Overhead, ISCAS '85 Circuits, SAED32nm HVT Library

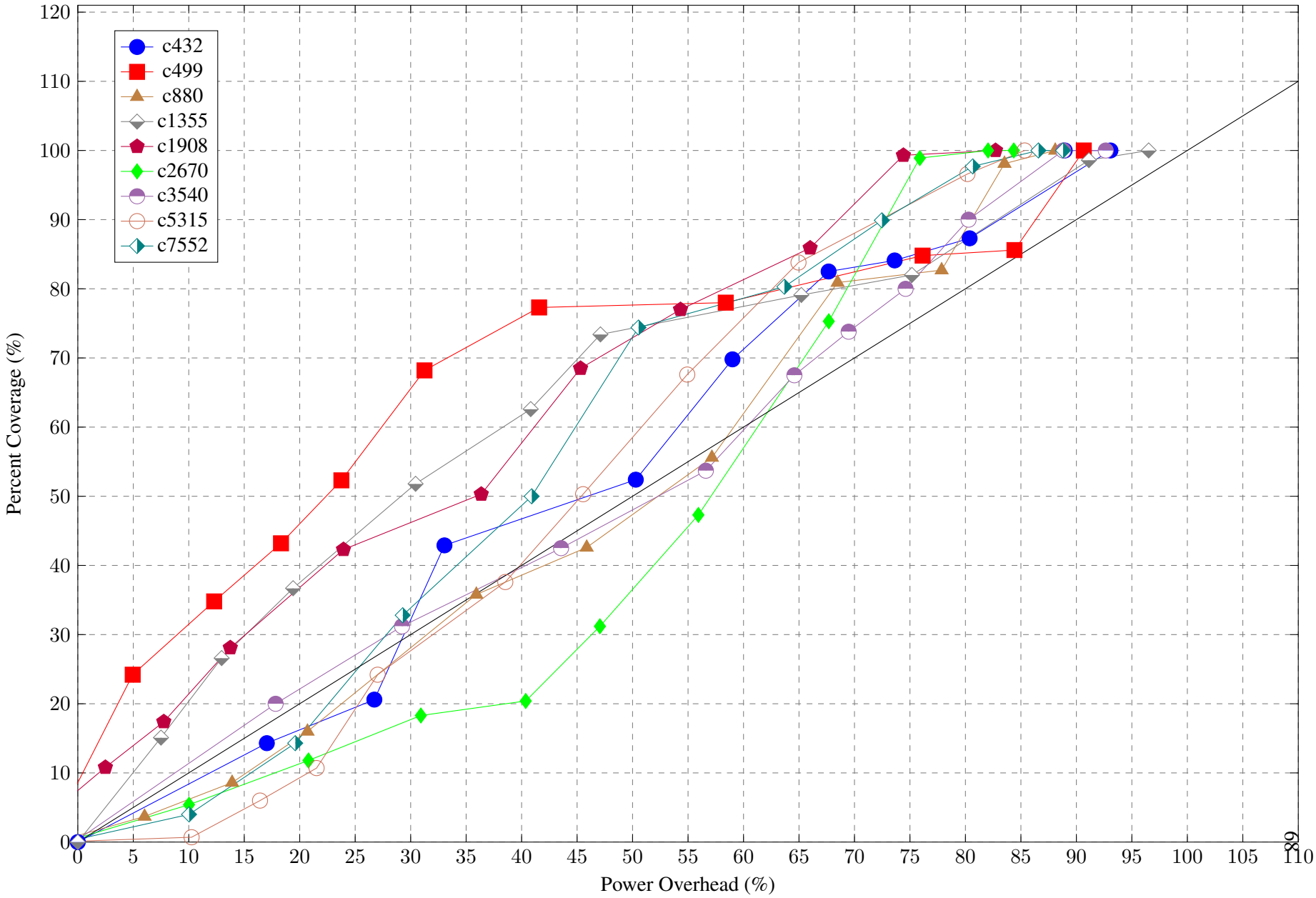


Figure 5.15: inSense Exoskeleton SET Error Detection Rate vs. Power Overhead, ISCAS '85 Circuits, SAED90nm HVT Library

Chapter 6

Conclusions and Future Work

Future technology nodes, new devices, and novel materials present unfamiliar challenges due to their complexity and increased sensitivity to process and environmental variations. Consequently, the development of reliable systems from unreliable components and materials is becoming a topic of increasing significance. The inSense Architecture is envisioned such that new designs can be realized with new technologies despite issues with reliability and device variation. The future of computing and electronics will require fundamental changes in the way that information is encoded and processed. While in the short-term this will entail new device structures and new materials, in the long term this may mean new information paradigms (*e.g.* spintronics or quantum computing) and novel logic system. Imbuing low-level hardware with sensory capabilities in order to regulate performance and enhance stability appears to be in line with these upcoming paradigm shifts.

The prototype implementation shows potential, especially when for some of the larger designs sizes; c2670, c5315, and c7552 of the ISCAS '85 benchmark suite deliver good error coverage with a reasonable amount of area overhead. The Soft-Error Detection Unit is responsible for the majority of the overhead, and the greatest gains would come from its improvement. From our literature examination, it seems that the preferred methodology would involve resizing of select gates/transistors, although this could be challenging as gate-resizing of nodes on the critical path would increase total delay. Results for partial duplication and gate-resizing could be improved through the use of published algorithms [47]

or through repeated simulation of typical workloads [8]. For the delay monitoring unit, implementation of time-borrowing delay monitors would reduce overheads, and implementation of a quantitative delay monitor would allow for detailed health input to an intelligent control system. Further efficiency improvements could result through higher granularity delay elements or custom cell implementations of delay monitors. Improved mitigation of within-die process variation could be realized through more sophisticated delay-monitor insertion strategies; for example, multiple slow paths per combinatorial logic block could be monitored. Various state-space search algorithms, such as k-means clustering, could be implemented in order to more precisely identify ideal DMU coverage given an overhead budget.

This work opens up many additional exciting possibilities for continuing research. As previously mentioned, improvement of each component of the inSense Exoskeleton would prove interesting research topics. Alternative DMU and SDU implementations could be explored and contrasted for various designs.

Completing the inSense Exoskeleton would necessitate the design of an efficient VCU along with an automated insertion process. The VCU could leverage existing instruction replay logic in microprocessor designs, or a custom control system could be designed. For designs with a large number of functional units such as multi-core processors, inSense-augmented blocks could use some type of inter-unit communication network and protocol to monitor overall system health and status, leveraging the research conducted in sensor mini-networks.

It would also be valuable to implement the inSense architecture on a large industry design, such as the OpenSPARC or LEON-3 processors. Targeting specific frequency/area/power goals and evaluating the inSense architecture for a practical design utilizing a predictive technology model could provide valuable insights toward further improvement.

Automation of the architecture could also be improved. The current process involves

many different scripts and tools which must be orchestrated in order to yield the desired result; further consolidation of the flow into Synopsys Design Compiler would streamline the process, increasing ease-of-use while decreasing the possibility of bugs and incompatibilities. Another possibility comprises the possibility of a domain-specific language allowing designers to specify intent regarding inSense (and other) sensory components; this is along the lines of capability ala the united power format (UPF) standard.

Finally, the architecture could be specifically tailored towards the needs of post-CMOS devices and materials. While the prototype inSense implementation is applicable to transistor designs using other semiconductor materials such as indium gallium arsenide or germanium rather than silicon, more radical departures from classical FETs such as carbon nanotubes may benefit from modified or completely different sensory and feedback components in the architecture.

Bibliography

[1] M.C. Hansen, H. Yalcin, and J.P. Hayes. Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering. *Design Test of Computers, IEEE*, 16(3):72 –80, 1999.

[2] K. Bernstein, D.J. Frank, A.E. Gattiker, W. Haensch, B.L. Ji, S.R. Nassif, E.J. Nowak, D.J. Pearson, and N.J. Rohrer. High-performance CMOS variability in the 65-nm regime and beyond. *IBM Journal of Research and Development*, 50(4.5):433–449, July 2006.

[3] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. Parameter variations and impact on circuits and microarchitecture. In *Proceedings of the Conference on Design Automation*, pages 338–342, 2003.

[4] J.W. Tschanz, J.T. Kao, S.G. Narendra, R. Nair, D.A. Antoniadis, A.P. Chandrakasan, and V. De. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. *IEEE Journal of Solid-State Circuits*, 37(11):1396–1402, Nov. 2002.

[5] Minal Sawant. Single event effects complicate military avionics system design. *Military Electronics and Computing, The Journal of*, Jan. 2012.

[6] Robert C Baumann. Radiation-induced soft errors in advanced semiconductor technologies. *Device and Materials Reliability, IEEE Transactions on*, 5(3):305–316, 2005.

[7] D.G. Mavis and P.H. Eaton. Soft error rate mitigation techniques for modern microcircuits. In *Proceedings of the Reliability Physics Symposium*, pages 216–225, 2002.

[8] K. Mohanram and N. A. Touba. Cost-effective approach for reducing soft error failure rate in logic circuits. In *Proceedings of the International Test Conference*, pages 893–901, September/October 2003.

[9] Agilent. Application note b1500-17.

[10] D. Sylvester, D. Blaauw, and E. Karl. Elastic: An adaptive self-healing architecture for unpredictable silicon. *IEEE Design & Test of Computers*, 23(6):484–490, June 2006.

[11] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D.M. Bull, and D.T. Blaauw. Tdtb_dm: In situ error detection and correction for pvt and ser tolerance. *IEEE Journal of Solid-State Circuits*, 44(1):32–48, Jan. 2009.

[12] M. Mehrara, M. Attariyan, S. Shyam, K. Constantinides, V. Bertacco, and T. Austin. Low-cost protection for SER upsets and silicon defects. In *Proceedings of the Design, Automation & Test Conference in Europe*, pages 1–6, April 2007.

[13] S. Gupta, Shuguang Feng, A. Ansari, J. Blome, and S. Mahlke. The stagenet fabric for constructing resilient multicore systems. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture*, pages 141–151, Nov. 2008.

[14] Mohammad Reza Kakoee, Ashoka Sathanur, Antonio Pullini, and Luca Benini. Row-based fbb: A design-time optimization for post-silicon tunable circuits. *Microelectronics Journal*, 43(7):456 – 465, 2012.

[15] K.A. Bowman, S.G. Duvall, and J.D. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *IEEE Journal of Solid-State Circuits*, 37(2):183–190, Feb 2002.

[16] K. Bernstein, DJ Frank, AE Gattiker, W. Haensch, BL Ji, SR Nassif, EJ Nowak, DJ Pearson, and NJ Rohrer. High-performance CMOS variability in the 65-nm regime and beyond. *IBM Journal of Research and Development*, 50(4-5):433–450, 2006.

[17] Tze-Chiang Chen. Where CMOS is going: trendy hype vs. real technology. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 1–18, Feb. 2006.

[18] International Roadmap Committee et al. International technology roadmap for semiconductors 2013 edition, ERD (emerging research devices).

[19] N. Patil, Jie Deng, S. Mitra, and H.-S.P. Wong. Circuit-level performance benchmarking and scalability analysis of carbon nanotube transistor circuits. *Nanotechnology, IEEE Transactions on*, 8(1):37 –45, jan. 2009.

[20] Y.-B. Kim. Challenges for nanoscale MOSFETs and emerging nanoelectronics. *Electrical and Electronic Materials, Transactions on*, 11(3):93–105, June 2010.

[21] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 389–398, 2002.

[22] J.M. Benedetto, P.H. Eaton, D.G. Mavis, M. Gadlage, and T. Turflinger. Variation of digital SET pulse widths and the implications for single event hardening of advanced cmos processes. *IEEE Transactions on Nuclear Science*, 52(6):2114–2119, Dec. 2005.

[23] B. Narasimham, M. J. Gadlage, B. L. Bhuva, R. D. Schrimpf, L. W. Massengill, W. T. Holman, A. F. Witulski, Xiaowei Zhu, A. Balasubramanian, and S. A. Wender. Neutron and alpha particle-induced transients in 90 nm technology. In *Proceedings of the International Reliability Physics Symposium*, pages 478–481, April/May 2008.

[24] M.J. Gadlage, R.D. Schrimpf, J.M. Benedetto, P.H. Eaton, D.G. Mavis, M. Sibley, K. Avery, and T.L. Turflinger. Single event transient pulse widths in digital microcircuits. *IEEE Transactions on Nuclear Science*, 51(6):3285–3290, Dec. 2004.

[25] B. Narasimham, B.L. Bhuva, R.D. Schrimpf, L.W. Massengill, M.J. Gadlage, O.A. Amusan, W.T. Holman, A.F. Witulski, W.H. Robinson, J.D. Black, J.M. Benedetto, and P.H. Eaton. Characterization of digital single event transient pulse-widths in 130-nm and 90-nm cmos technologies. *IEEE Transactions on Nuclear Science*, 54(6):2506–2511, Dec. 2007.

[26] V. Chandra and R. Aitken. Impact of technology and voltage scaling on the soft error susceptibility in nanoscale CMOS. In *Proceedings of the IEEE International*

Symposium on Defect and Fault Tolerance of VLSI Systems, pages 114–122, Oct. 2008.

[27] B Gill, N Seifert, and V Zia. Comparison of alpha-particle and neutron-induced combinational and sequential logic error rates at the 32nm technology node. In *Reliability Physics Symposium, 2009 IEEE International*, pages 199–205. IEEE, 2009.

[28] DF Heidel, KP Rodbell, EH Cannon, C. Cabral Jr, MS Gordon, P. Oldiges, and HHK Tang. Alpha-particle-induced upsets in advanced CMOS circuits and technology. *IBM Journal of Research and Development*, 52(3):225–231, 2008.

[29] Tino Heijmen. Soft errors from space to ground: Historical overview, empirical evidence, and future trends. In Michael Nicolaidis, editor, *Soft Errors in Modern Electronic Systems*, volume 41 of *Frontiers in Electronic Testing*, pages 1–25. Springer US, 2011.

[30] P.E. Dodd, M.R. Shaneyfelt, J.R. Schwank, and J.A. Felix. Current and future challenges in radiation effects on CMOS electronics. *Nuclear Science, IEEE Transactions on*, 57(4):1747 –1763, aug. 2010.

[31] V. Reddy, A.T. Krishnan, A. Marshall, J. Rodriguez, S. Natarajan, T. Rost, and S. Krishnan. Impact of negative bias temperature instability on digital circuit reliability. *Microelectronics Reliability*, 45(1):31–38, 2005.

[32] International Roadmap Committee et al. International technology roadmap for semiconductors 2013 edition, PIDS (process integration, devices, and structures).

[33] T. Chen and S. Naffziger. Comparison of adaptive body bias (ABB) and adaptive supply voltage (ASV) for improving delay and leakage under the presence of process variation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(5):888–899, Oct. 2003.

[34] Radu Teodorescu, Jun Nakano, Abhishek Tiwari, and Josep Torrellas. Mitigating parameter variation with dynamic fine-grain body biasing. In *Proceedings of the International Symposium on Microarchitecture*, pages 27–42, 2007.

[35] M Lauriente and AL Vampola. Spacecraft anomalies due to radiation environment in space. *NASA Goddard Space Flight Center, Greenbelt, MD*, 20771, 1996.

[36] D. Bessot and R. Velazco. Design of SEU-hardened CMOS memory cells: the HIT cell. In *Second European Conference on Radiation and its Effects on Components and Systems*, pages 563–570, Sep 1993.

[37] Jr. Rockett, L.R. An SEU-hardened CMOS data latch design. *IEEE Transactions on Nuclear Science*, 35(6):1682–1687, Dec 1988.

[38] A. Balasubramanian, B.L. Bhuva, J.D. Black, and L.W. Massengill. RHBD techniques for mitigating effects of single-event hits using guard-gates. *IEEE Transactions on Nuclear Science*, 52(6):2531–2535, Dec. 2005.

[39] M. Nicolaidis. Design for soft error mitigation. *IEEE Transactions on Device and Materials Reliability*, 5(3):405–418, Sept. 2005.

[40] E. Rotenberg. Ar-smt: A microarchitectural approach to fault tolerance in microprocessors. In *Proceedings of the International Symposium on Fault-Tolerant Computing*, pages 84–91, 1999.

[41] S.K. Reinhardt and S.S. Mukherjee. Transient fault detection via simultaneous multithreading. In *Proceedings of the International Symposium on Computer Architecture*, pages 25–36, 2000.

[42] Vimal K. Reddy, Eric Rotenberg, and Sailashri Parthasarathy. Understanding prediction-based partial redundant threading for low-overhead, high- coverage fault tolerance. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 83–94, 2006.

[43] K. Mohanram and N.A. Touba. Partial error masking to reduce soft error failure rate in logic circuits. In *Proceedings of the International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 433–440, Nov. 2003.

[44] A.J. KleinOsowski, K. KleinOsowski, V. Rangarajan, P. Ranganath, and D.J. Lilja. The recursive nanobox processor grid: a reliable system architecture for unreliable nanotechnology devices. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 167–176, June-1 July 2004.

[45] M. Miyazaki, G. Ono, and K. Ishibashi. A 1.2-gips/w microprocessor using speed-adaptive threshold-voltage cmos with forward bias. *IEEE Journal of Solid-State Circuits*, 37(2):210–217, Feb 2002.

[46] K.A. Bowman, J.W. Tschanz, Nam Sung Kim, J.C. Lee, C.B. Wilkerson, S.-L.L. Lu, T. Karnik, and V.K. De. Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance. *Solid-State Circuits, IEEE Journal of*, 44(1):49 –63, jan. 2009.

[47] Natasa Miskov-Zivanov and Diana Marculescu. MARS-C: modeling and reduction of soft errors in combinational circuits. In *Proceedings of the 43rd annual Design Automation Conference*, DAC '06, pages 767–772, New York, NY, USA, 2006. ACM.

[48] Bin Zhang, Wei-Shen Wang, and Michael Orshansky. FASER: Fast analysis of soft error susceptibility for cell-based designs. In *Proceedings of the 7th International Symposium on Quality Electronic Design*, ISQED '06, pages 755–760, Washington, DC, USA, 2006. IEEE Computer Society.

[49] Synopsys 90nm generic library for teaching ic design, June 2010.

[50] Synopsys 32/28nm generic library for teaching ic design, December 2012.

[51] Kevin Kleinosowski, Aj Kleinosowski, and David J. Lilja. The SEASONing Tool: A Spice Engine for Adding Soft-errors On Netlists. 2008.

[52] S. Borkar. Designing reliable systems from unreliable components: The challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, Nov./Dec. 2005.

[53] L. Benini and G. De Micheli. Networks on chips: a new soc paradigm. *Computer*, 35(1):70–78, Jan 2002.

[54] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C.R. Das. Exploring fault-tolerant network-on-chip architectures. pages 93–104, June 2006.

[55] D.J. Frank, R.H. Dennard, E. Nowak, P.M. Solomon, Y. Taur, H.S.P. Wong, et al. Device scaling limits of Si MOSFETs and their application dependencies. *PROCEEDINGS-IEEE*, 89(3):259–288, 2001.

[56] W. Haensch, EJ Nowak, RH Dennard, PM Solomon, A. Bryant, OH Dokumaci, A. Kumar, X. Wang, JB Johnson, and MV Fischetti. Silicon CMOS devices beyond scaling. *IBM Journal of Research and Development*, 50(4-5):339–362, 2006.

[57] Synopsys leda user guide version e-2010.12, December 2010.